Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Dohledový a konfigurační systém pro podnikovou síť

Monitoring and Configuring System for Corporate Network

2015 Bc. Zdeněk Nábělek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Zdeněk Nábělek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2601T013 Telekomunikační technika

Téma: Dohledový a konfigurační systém pro podnikovou síť
Monitoring and Configuring System for Corporate Network

Zásady pro vypracování:

1. Popište současné možnosti správy počítačové sítě pomocí vzdáleného dohledu.
2. Navrhněte komplexní dohledový a konfigurační systém pro vzdálenou správu podnikové počítačové sítě. Dohledový systém musí být přístupný přes webové rozhraní.
3. Navržený systém implementujte do reálného prostředí počítačové sítě.
4. Proveďte ověření funkčnosti systému řadou specifických testů.

Seznam doporučené odborné literatury:

LECKY-THOMPSON, Ed a Steven D NOWICKI. PHP 6: programujeme profesionálně. Vyd. 1. Překlad Ondřej Gibl. Brno: Computer Press, 2010, 718 s. Programujeme profesionálně. ISBN 978-80-251-3127-5.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

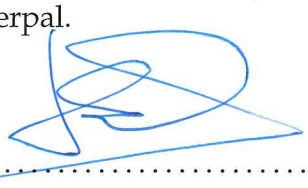
Vedoucí diplomové práce: **Ing. Libor Michalek, Ph.D.**

Datum zadání: 01.09.2014
Datum odevzdání: 07.05.2015

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.



V Ostravě 7. května 2015 .....

Chtěl bych poděkovat svému vedoucímu diplomové práce Ing. Liboru Michalkovi, Ph.D. za cenné rady a připomínky k mé práci. Mé podekování patří též mé rodině a blízkým přátelům za pomoc a podporu během studia.

Abstrakt

Cílem této práce je vyvinout program, který má sjednotit a usnadnit každodenní činnosti pracovníků dohledového centra. Tento program je současně napojen na další systémy, nahrazuje práci s nimi a využívá několika typů databází. Výsledný program rozšiřuje základní operace o specifické úpravy, které byly vyvinuty speciálně pro potřeby směnového provozu, a je uživateli prezentován v ucelené formě jako webová stránka. K jeho vývoji bylo použito skriptovacího jazyka PHP, javascriptu a tří typů databází Oracle, MsSQL a MySQL.

Klíčová slova: Monitorování, dohledové nástroje, aktivní síťové prvky, Zabbix, PHP, MRTG

Abstract

The goal of this master thesis is to develop a program to unify and facilitate the daily activities of staff supervision center. This program is also connected to other systems, replaces work with them and uses several types of databases. The resulting program extends the basic operations of specific adjustments that have been developed specifically for the needs of shift operation and is presented to the user in a compact form as a web page. In order to develop the program scripting language PHP, javascript and three types of databases (Oracle, MsSQL, MySQL) were used.

Keywords: Monitoring, surveillance tools, active network elements, Zabbix, PHP, MRTG

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface - označuje v informatice rozhraní pro programování aplikací.
BRI	–	Basic Rate Interface - modul v routeru umožňující připojení ISDN.
ČP	–	Česká pošta.
DB	–	Databáze.
DNS	–	Domain Name System - hierarchický systém doménových jmen.
DSL	–	Digital Subscriber Line - technologie, která umožňuje využít stávající vedení telefonu nebo kabelové televize.
GIS	–	Geographic information systém - geografický informační systém.
IP Sec	–	Internet Protocol Security - je bezpečnostní rozšíření IP protokolu založené na autentizaci a šifrování každého IP datagramu.
IPMI	–	Intelligent Platform Management Interface - poskytuje možnosti vzdáleného přístupu, monitorování a administrace serverů a dalšího hardware.
ISDN	–	Integrated Services for Digital Network - digitální síť integrovaných služeb.
ISP	–	Internet service provider - poskytovatel internetového připojení.
JMX	–	Java Management Extensions - poskytuje nástroje pro vytváření webových aplikací, správu a monitorování síťových zařízení.
LDAP	–	Lightweight Directory Access Protocol - je definovaný protokol pro ukládání a přístup k datům na adresářovém serveru.
MRTG	–	Multi Router Traffic Grapher - je software na malování grafů podle SNMP veličin.

NE	–	Network element - síťové zařízení.
NI	–	Network Intelligence - zkrácený název programu.
CNI	–	CROSS Network Intelligence - název programu.
NT	–	Next technology - další typy technologií použité pro konektivitu.
O2	–	Telefonica O2 - firma poskytující konektivitu a další služby.
OID	–	Object identifier - ve výpočetní technice je identifikátor sloužící k jednoznačné identifikaci objektu.
OSS	–	Open source software - je počítačový software s otevřeným zdrojovým kódem.
PHP	–	Server-side scripting language - skriptovací jazyk pro programování.
ROI	–	Return on investment - znamená poměr výnosu či změny hodnoty investice k investovanému kapitálu.
SGML	–	Standard Generalized Markup Language - je univerzální značkovací metajazyk, který umožňuje definovat značkovací jazyky jako své vlastní podmnožiny.
SNMP	–	Simple Network Management Protocol - umožňuje průběžný sběr nejrůznějších dat pro potřeby správy sítě.
SSH	–	Secure Shell - je v informatice označení pro program a zároveň pro zabezpečený komunikační protokol v počítačových sítích, které používají TCP/IP.
TS	–	Timeslot - kanál o základní přenosové rychlosti 64 kbit/s.
WWW	–	World Wide Web - světová rozsáhlá síť.
XML	–	Extensible Markup Language - rozšiřitelný značkovací jazyk.

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Softwarové vybavení dohledových středisek se neustále vyvíjí kupředu a má mnoho podob. Jednotlivá řešení nabízí spousta firem s odlišnými způsoby implementace a s různým zaměřením. Zcela logicky lze chápat, že dohledové středisko pro kamerový systém bude vypadat odlišně v porovnání s dohledovým centrem aplikací či aktivních síťových prvků.

V praxi se setkáme se situací, kdy se většinou kombinuje více aplikací pro širší záběr dohledu. To přináší své výhody i nevýhody. Každý takto využívaný program je s největší pravděpodobností odladěn a zaměřen na konkrétní část. Jeho výstupy představují přesně to, co se od daného programu čekává. Na druhou stranu se musí pracovníci školit na každý program zvlášť a taky se musí naučit jaké informace lze z dané aplikace získat. Tato práce má obdržené výstupy ucelit, zpřehlednit a poskytnut je na jednom místě. Výsledkem tedy bude jeden program, který zrychlí a zefektivní práci zaměstnance dohledového centra.

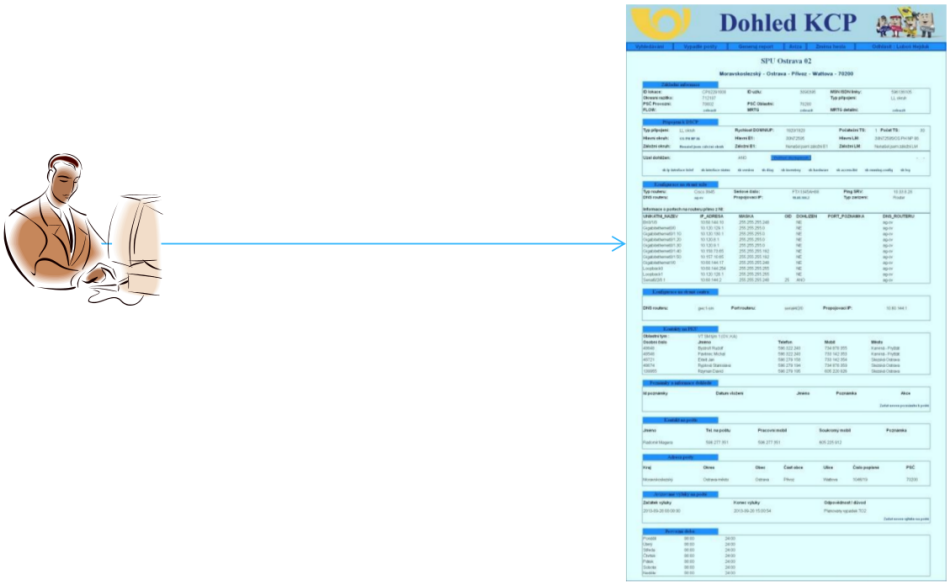
Vzhledem k mému zaměření a možnostem jsem zvolil dohledové centrum zabývající se provozem datové sítě. Mezi nejznámější bezplatné programy, které se pro tyto účely využívají bezpochyby patří Nagios, Catci a Zabbix. V komerční sféře často pracuje s placenými programy jako jsou Zenoss, PacketTrap pt360 nebo známější HP OpenView či CiscoWorks LAN Management Solution.

V této práci se bude pracovat s několika podobnými programy, datovými modely a aktivními prvky. K výše zmíněným programům přibude Service desk, Monitorování aktivních prvků, interní aplikace dohledu ČP a online práce s routery. Vyvinutá aplikace pomůže při generování reportů a při plánování předem známých akcí. Její přidané hodnoty jsou shrnuty do následujících bodů:

- Předávání informací ke konkrétním prvkům.
- Plánování a přehled výluk.
- Online test dostupnosti.
- Interaktivní přístup a výpisy.
- Přehled vypadlých aktivních prvků.
- Generování XML.
- Sumarizace tiketů.
- Úprava portů síťových zařízení.

Výše uvedené přidané hodnoty lépe vystihuje následující grafika (obr. 1), (obr. 2).

Obrázek 1: Původní schéma přístupu pracovníka dohledu



Obrázek 2: Cílový stav

2 Základní popis

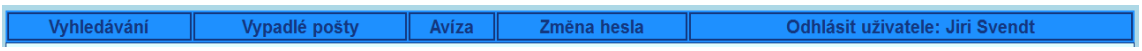
Pro prezentaci informací koncovému pracovníkovi aplikace využívá jak několika zdrojů, tak přístupů k různým funkcionalitám. Jelikož v současné době aplikaci využívá jen devět zaměstnanců bylo zvoleno umístění přihlašovacích údajů v lokální databázi. Vzhledem ke struktuře programu a rozvržení do tříd lze jednoduše tento přístup zaměnit třeba za LDAP.

2.1 Přihlášení uživatele

S ohledem na dostupné prostředky v aplikaci byly přístupy uživatelů rozděleny do dvou základních skupin:

- Administrátor (obr. 3).
- Uživatel (obr. 4).

Obrázek 3: Rozhraní administrátora



Obrázek 4: Rozhraní uživatele

2.2 Vyhledávání

Důležitým aspektem pro získaní informací je vhodně zvolené klíčové slovo. Pro větší usnadnění bylo zvoleno vyhledávání podle kritérií jako je lokalita, název síťového zařízení, IP adresa, PSČ či název přípojeného okruhu (obr. 5). Pracovník dostává základní informace ihned při prvotním výpisu, kdy jen upřesní, o který konkrétně nalezený údaj má zájem. Například s lokalitou Opava je spojeno celkem devět síťových zařízení (obr. 6). Operátor má ihned přehled o jejich názvech, umístění, propojovací IP adrese, názvu propojovacích okruhů a jejich rychlostí. Po kliknutí na požadovaný záznam je zobrazena detailní informace o nalezeném síťovém zařízení, která bude v práci popsáná dále.

Obrázek 5: Ukázka typu kategorii pro vyhledávání

ID_LOKALITY	DNS_ROUTERU	POSTA	ULICE	PSC	IP_ADRESA	RYCHLOST	NAZEV_OKRUHU
CP04094000	ag-p707777	depo Opava 70	Podvihovská	74770	10.224.232.242	4096	OPAV OPAV LSSDSL 800140
CP02200000	ag-op	obvod Opava	Masarykova třída	74601	10.60.136.2	512	OPAV PH NP 10
CP02196000	ag-p707025	pošta Opava 2	Husova	74601	10.61.55.2	128	OPAV PH NP 3
CP02201000	ag-p707052	pošta Opava 5	Partyzánská	74705	10.61.56.2	128	OPAV PH NP 4
CP02202000	ag-p707070	pošta Opava 7	Přemyslovců	74707	10.61.58.2	64	OPAV PH NP 5
CP02199000	ag-p707294	pošta Opava 9	Na Spojce	74770	10.61.59.2	128	OPAV PH NP 6
CP02193000	ag-p707061	pošta Opava 6	Hlavní	74706	10.61.57.2	128	OPAV PH NP 7
CP02203000	ag-p707688	pošta Opava 8	Těšínská	74601	10.62.77.2	64	OPAV PH NP 8
CP04271000	ag-p707016	pošta Opava 1	U Fortny	74601	10.3.60.1	10240	PH POŠTA 032 OPAVA 1

Obrázek 6: Příklad výpisu nalezených objektů pro uzel s názvem Opava

3 Databáze

Databáze obecně slouží pro ukládání dat číselných, textových nebo v jiné podobě. Práce s informacemi, které jsou v ní uchovány, je prováděna na základě dotazů. Rozeznáváme čtyři základní typy dotazů. Pokud potřebujme data jen získat, pak lze použít typ SELECT. Nebudou-li již určitá data potřeba a chtěli bychom je vymazat, využijeme dotaz typu DELETE. Vložení nového záznamu se provede za pomoci typu INSERT. Poslední možností je situace, kdy potřebujeme data změnit. K tomuto účelu slouží typ UPDATE. Všechny tyto typy mají podobnou strukturu. Je nutno říci, který typ se použije, odkud se bude čerpat - FROM. Dále může být dotaz upřesněn nepovinnou podmínku - WHERE.

Rozlišujeme několik typů databází v závislosti na systému. V této práci budeme pracovat se třemi typy: ORACLE, MySQL a MsSql. Pro každý z těchto typů databáze byla vytvořena třída a funkce pro práci s ní. Jak taková třída vypadá a jaké volání umožňuje je uvedeno v příloze (výpis 18).

3.1 Ukázka dotazování do databáze

K získaní potřebných výsledných dat se využívá spojování různých tabulek na základě znalosti pole, které obsahuje identický prvek. Ukážeme si spojení PHP a SQL dotazu. Stavba takového dotazu bude vždy stejná, ale hledaný výraz se může měnit. Uveďme si konkrétní příklad na hledání "obec". Ve třídě PostAdres.php se vyhledá poštovní adresa a název pošty na základě zadaného výrazu, tj. obec. Z formuláře pro vyhledávání, který byl popsán v kapitole 2.2, získáme hledané slovo, které si uložíme do proměnné $object a tu následně předáme do dotazu. Výsledkem bude sestavený dotaz s konkrétním údajem, [12].

```
SELECT
    title , adresa
FROM
    OBJEKTM1
        LEFT JOIN DEVICE2M1 ON OBJEKTM1.LOGICAL_NAME=DEVICE2M1.
            LOGICAL_NAME
        LEFT JOIN LOCM1 ON DEVICE2M1.LOCATION=LOCM1.LOCATION
        LEFT JOIN SM_RECO.dbo.ppp_posty ON OBJEKTM1.POSTALCODE=
            SM_RECO.dbo.ppp_posty.psc
WHERE
    OBJEKTM1.LOGICAL_NAME ="'␣.␣$object␣.␣'"
```

Výpis 1: SQL dotaz typu SELECT

4 Interaktivní napojení na další využívané SW

V každém odvětví někdy nastane situace, kdy klasické informace, které jsou v běžném provozu dostačující, nemusí v danou chvíli stačit. Díky tomu, že jsou vyspecifikované jmenné konvence a striktně se dodržují, zaručuje to, že aktivní síťový prvek s názvem XXX najdeme ve všech využívaných programech se stejným názvem, tj. XXX. Proto byly na vybraných místech implementovány propojení do zdrojových aplikací na základě zobrazených informací. To znamená, že pracovník dohledového centra nemusí explicitně vyhledávat doplňkové informace. Aplikace zajistí interním předáním parametrů, získání veškerých požadovaných informací, které jsou prezentovány uživateli.

4.1 Cross Network Intelligence

CROSS NETWORK INTELLIGENCE (CNI, dále zkráceně jen NI) je vedlejší firmou Czech software developer a System Integrator SDC s.r.o. s více než desetiletou průmyslovou praxí zaměřující se na operační podpůrné systémy pro telekomunikační operátory. SDC vyvinul CROSS. CROSS je další generace síťového inventáře na bázi OSS se zaměřením na provozní flexibilitu, jednoduchost použití, zkrácení doby uvedení na trh pro nové služby a optimalizaci okamžité ROI.
CROSS je nezávislý na určité technologii, databázi nebo GIS, [3].

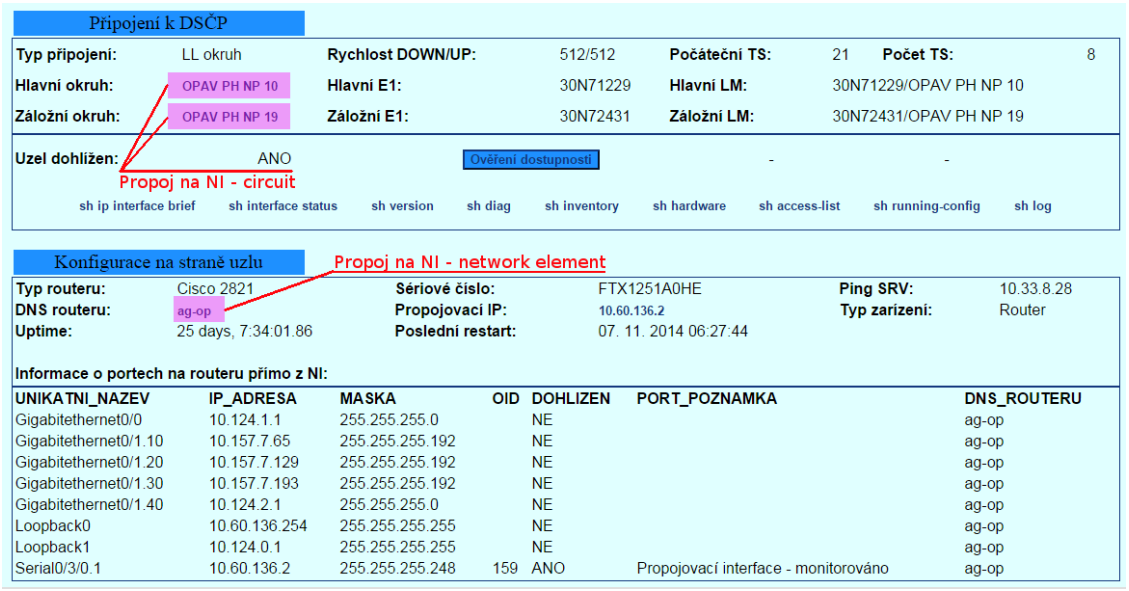
Propojení aplikace s NI je vytvořeno na základě názvu okruhu nebo DNS zařízení. Proto se v detailním zobrazení pro každý uzel na dvou místech objevuje propoj do tohoto systému (obr. 7). Sestavení odkazu pro NI probíhá získáním identifikačního údaje pro každou z těchto položek (výpis 2, 3). Po kliknutí je ihned k dispozici konkrétní okno pro okruh Circuit(obr. 8) respektive síťový prvek (obr. 9) a pracovník již nemusí nic dalšího hledat. Tyto propoje byly provedeny pro potřeby zpětné kontroly a zapisování změn v případě, že jsou zjištěny nesrovnalosti.

```
https://ni.cpost.cz/circ/index.htm#?circ_id=1234567&req_id=&gp_id=&sys_id=&v=
    default
```

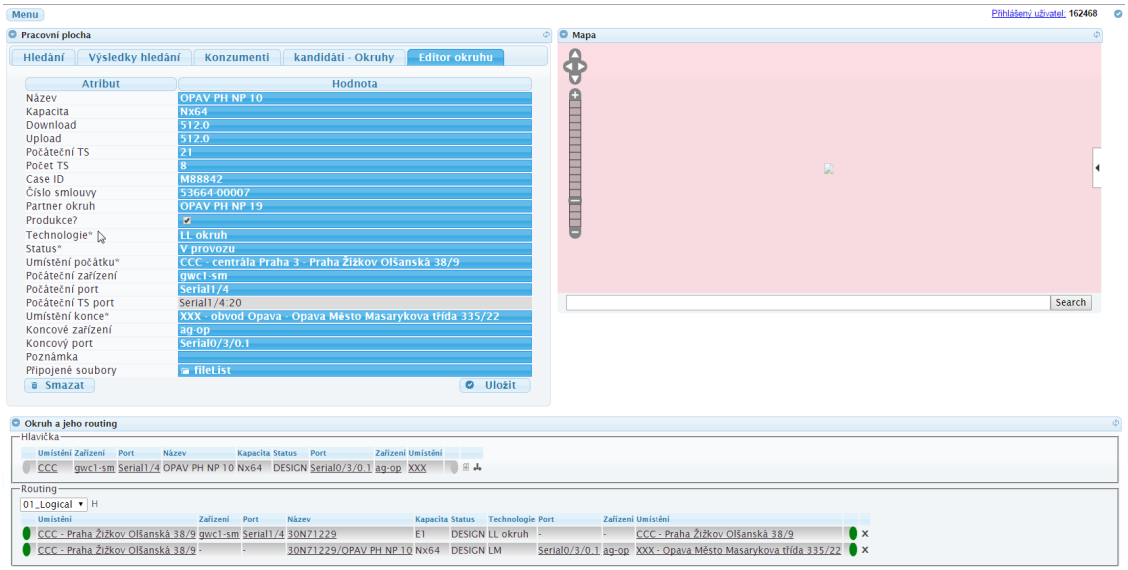
Výpis 2: Sestavený odkaz pro okruh do NI

```
https://ni.cpost.cz/loc/index.htm#?v=Edit&node_id=9876543
```

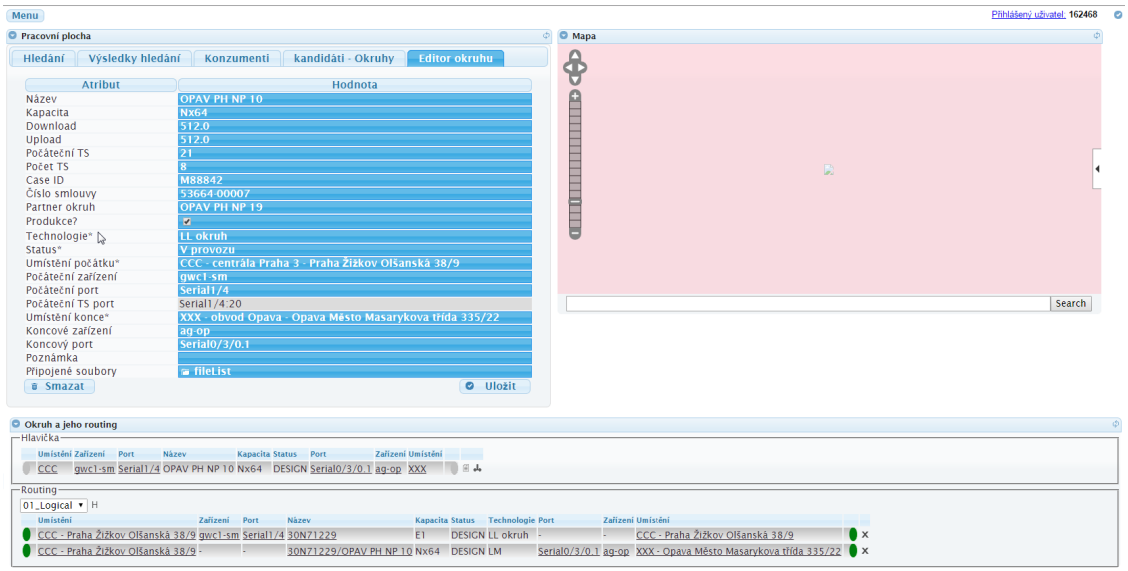
Výpis 3: Sestavený odkaz pro síťový prvek do NI



Obrázek 7: Umístění linku na propojení do aplikace Network Inventory



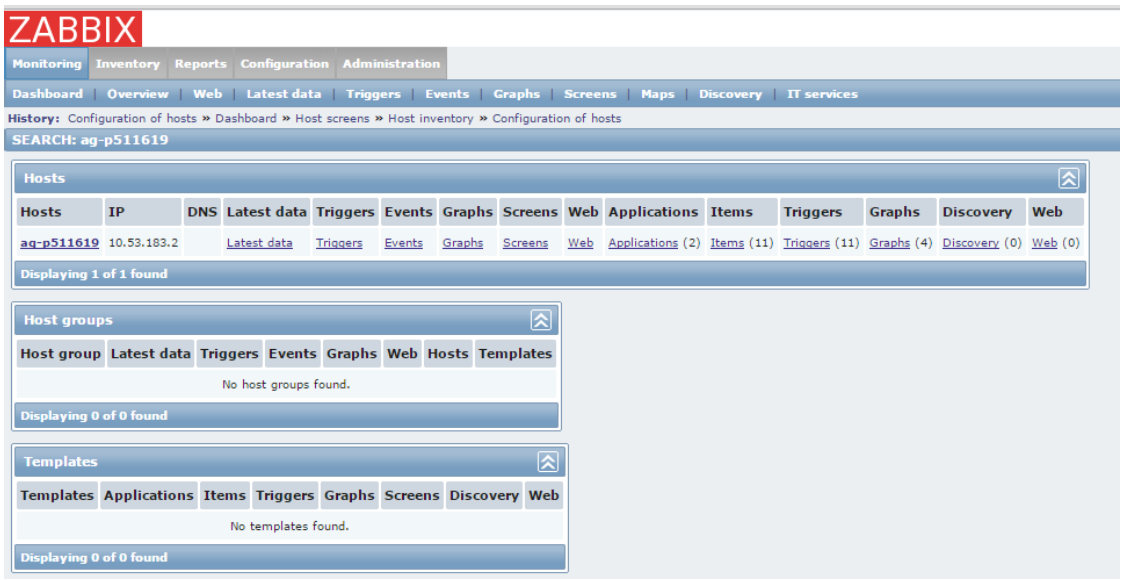
Obrázek 8: Ukázka výpisu okruhu



Obrázek 9: Ukázka výpisu NE

4.2 Zabbix

Zabbix slouží k monitorování aktivních síťových prvků (PC, servery, tiskárny, modemy, switche, UPS, ...), které jsou připojeny do počítačové sítě. Můžeme tedy sledovat stav a sbírat různé informace o všem, co má IP adresu. Metody pro sledování a zjišťování informací jsou různé. Mezi jednodušší metody patří například ICMP echo request. K složitějším metodám náleží SNMP , IPMI , JMX. Ke sledování lze použít také SSH, Telnet nebo agenta, který je dostupný pro většinu dnes používaných operačních systémů. Pomocí agenta je možné monitorovat nejen informace o stavu hardware (operační paměť, procesor, úložné zařízení,...), ale i systémové informace a stav běžících služeb. Dále je možné do prostředí integrovat vlastní externí skripty nebo s využitím API vytvořit vlastní testy. Podle tvůrců aplikace Zabbix lze monitorovat přes 100 000 prohlížených zařízení a provádět tak 1 000 000 vyhodnocení za minutu, což je závislé na systémových zdrojích serveru, na kterém je dohledový systém provozovaný. Zabbix může pracovat distribuovaně, to znamená, že v různých vzdálených lokalitách běží Zabbix v režimu proxy a data se následně přenáší na centrální server. To je vhodné pro velmi robustní a rozsáhlé sítě s velkým počtem zařízení. Dohledový systém je přístupný z webového rozhraní, které slouží zároveň i jako administrační prostředí pro správu a vyhodnocení dat, [5].

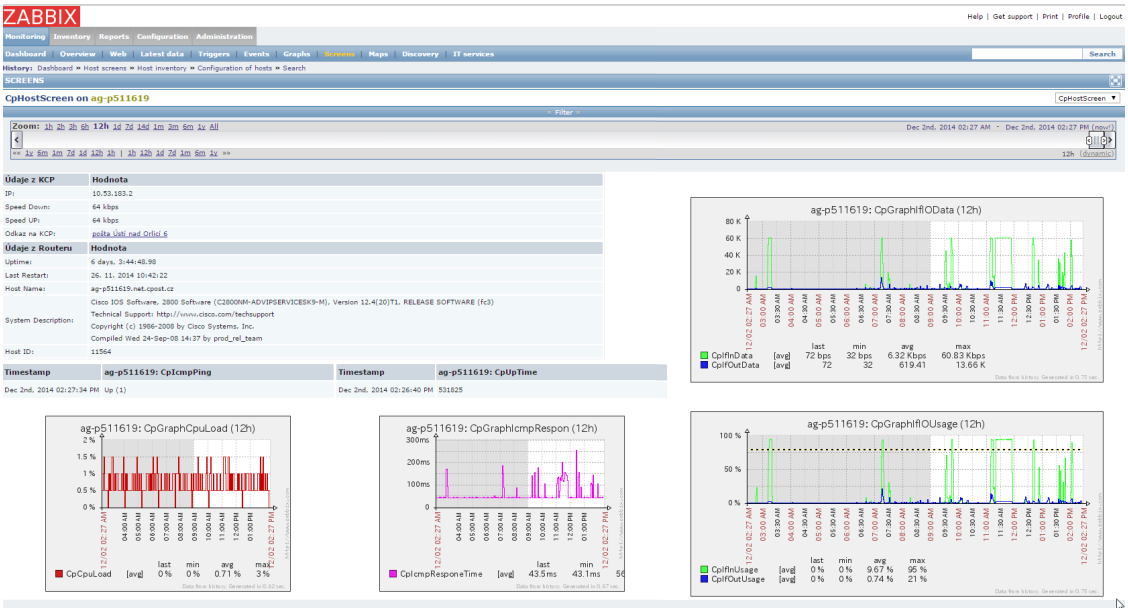


Obrázek 10: Základní obrazovka konkrétního uzlu

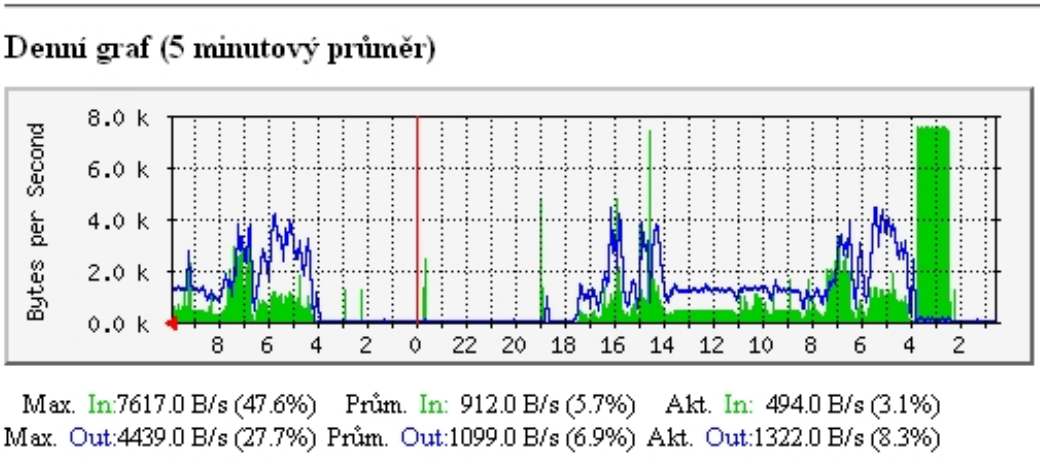
Aplikace Zabbix je dále také používaná v kapitole 6.1 k popisu zpracování souboru XML.

4.3 MRTG

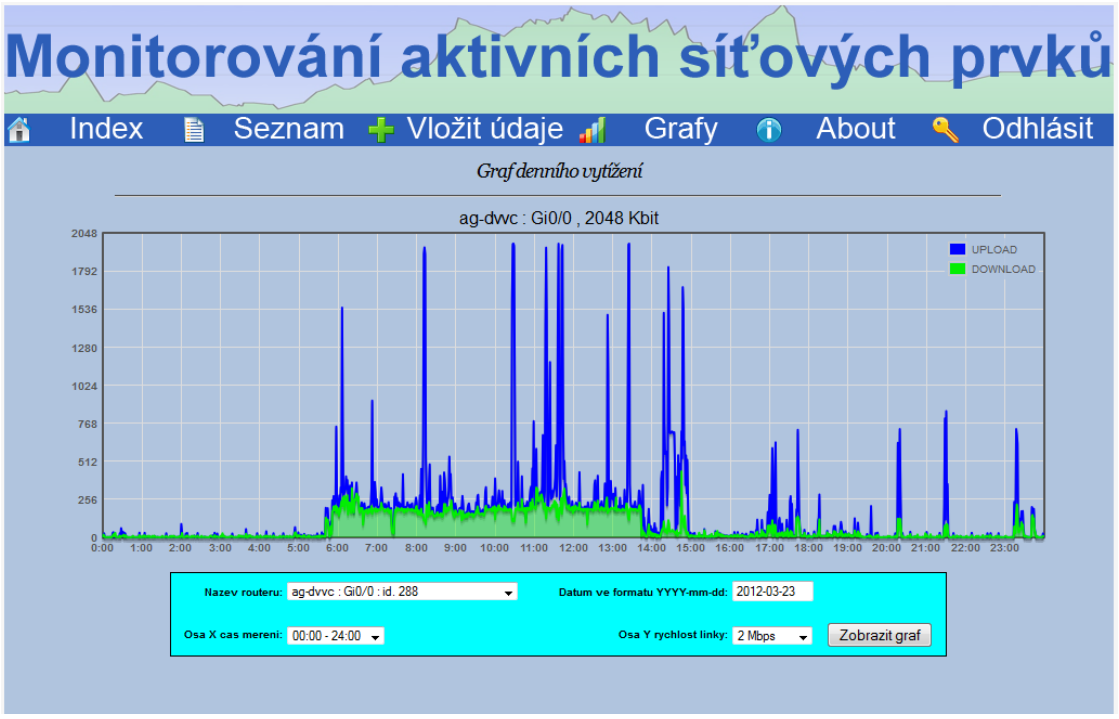
Dalším programem, který se využívá pro měření datového toku síťových prvků, je MRTG. Známe jej ve více verzích. V našem případě se konkrétně využívají celkem dvě respektive tři verze. Jedna z variant MRTG je výše zmíněný Zabbix (obr. 11). V prostředí, kde byla výsledná aplikace implementována, nastala situace, kdy z historických důvodů byl z části nasazen MRTG nástroj v základní verzi využívající perl skriptů (obr. 12). Bohužel tato varianta umožňuje dohlížet jen aktivní síťové prvky, které jsou do centrální sítě propojeny pomocí sériového rozhraní. Toto řešení vystačovalo do doby, než se začaly hromadně nasazovat technologie připojení typu VPN, IPSec nebo klasické ethertnet připojení. Proto byla v dříve k těmto účelům vypracována bakalářská práce, tedy program vycházející z funkcionality MRTG, ale pracující bez ohledu na typ konektivity (obr. 13). Umístění propojů je znázorněno na obrázku 14, [10].



Obrázek 11: Detailní zobrazení statistik konkretního uzlu



Obrázek 12: Ukázka dat při prokliku na ”staré”MRTG



Obrázek 13: Ukázka výstupu z aplikace Monitorování aktivních síťových prvků

Obrázek 14: Rozmístění linků na externí aplikace MRTG

4.4 NetFlow

Aplikace NetFlow je o otevřený protokol vyvinutý společností Cisco Systems, který byl původně určený jako doplňková služba k Cisco směrovačům. Jeho hlav-

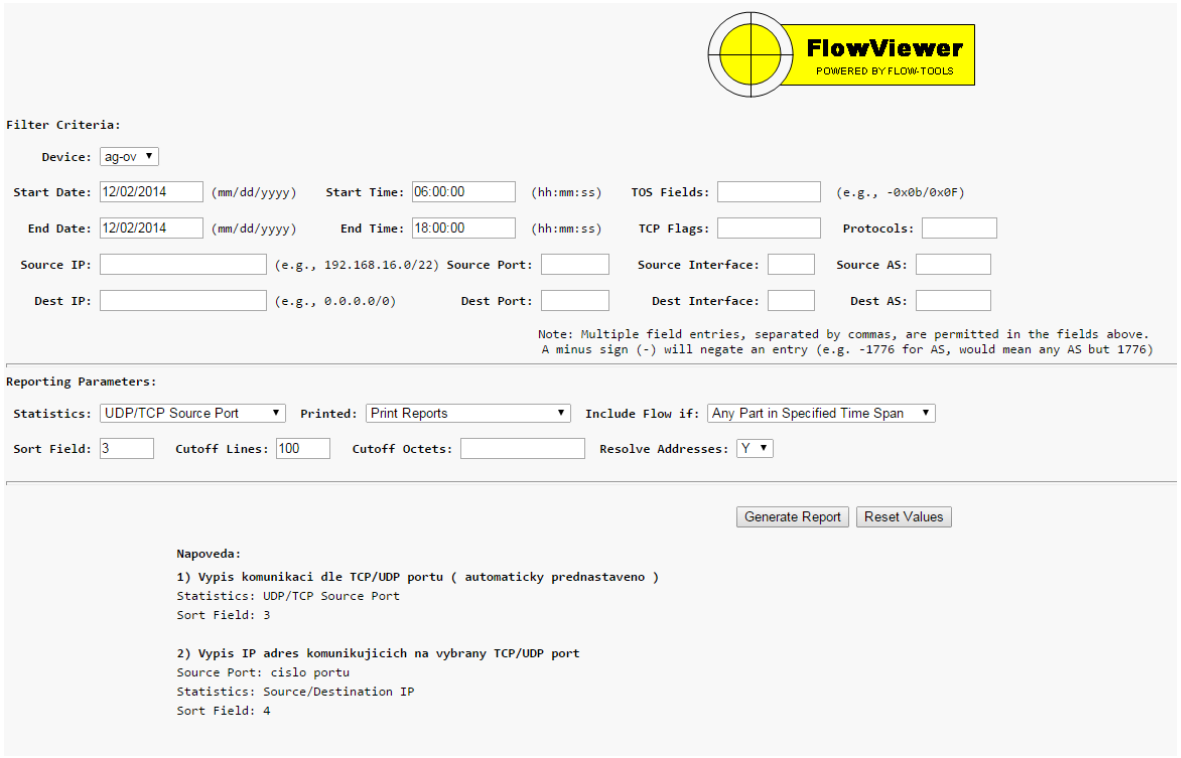
ním účelem je monitorování síťového provozu na základě IP toků. NetFlow poskytuje administrátorům i manažerům podrobný pohled do provozu na jejich síti v reálném čase. Proto tvoří důležitou a nepostradatelnou součást zabezpečení každé počítačové sítě a ISP na základě NetFlow statistik mohou svým zákazníkům účtovat ceny služeb v závislosti na množství přenesených dat. S pomocí NetFlow statistik lze odhalovat vnější i vnitřní incidenty, úzká místa v síti, dominantní zdroje provozu, efektivněji plánovat budoucí rozvoj sítě či sledovat, kdo s kým komunikoval, jak dlouho a s pomocí kterého protokolu, [1].

Dříve uvedené aplikace poskytovaly informace o prvcích ihned. Propoje do výše zmíněných aplikací jsou tedy jen z důvodů přístupu ke grafům. K této aplikaci budeme ale přistupovat trochu odlišně. Je to dáno jejími vlastnostmi a kapacitou. Vzhledem k hardwarovým nárokům může být v současné bodě na kolektoru zapnuto jen 30 sledovaných uzlů. Proto není vhodné automaticky odkazovat do aplikace NetFlowView a trvale předávat název síťového prvku. Docházelo by k nefunkčním propojům a interaktivita by byla zavádějící a matoucí. Tudíž byl vymyšlen způsob, jak tvorbu odkazů zautomatizovat, ale předávat pouze funkční linky. Tohoto bylo docíleno pomocí skriptu (výpis 4), který se přihlásí na server, kde běží netflow kolektor a zjistí, zda kontrolovaný uzel má aktivní měření. Zároveň provede kontrolu zapnutí i na patřičném síťovém prvku. Pokud je na obou místech měření aktivní, uživatel je po kliknutí na odkaz přesměrován rovnou na jeho výstup (obr. 15). V opačném případě se objeví informativní zpráva ”Na tomto uzlu není flow aktivní”. Jak vypadá výstup komunikace z pohledu portů na síťovém prvku za určité období můžeme vidět na obrázku 16.

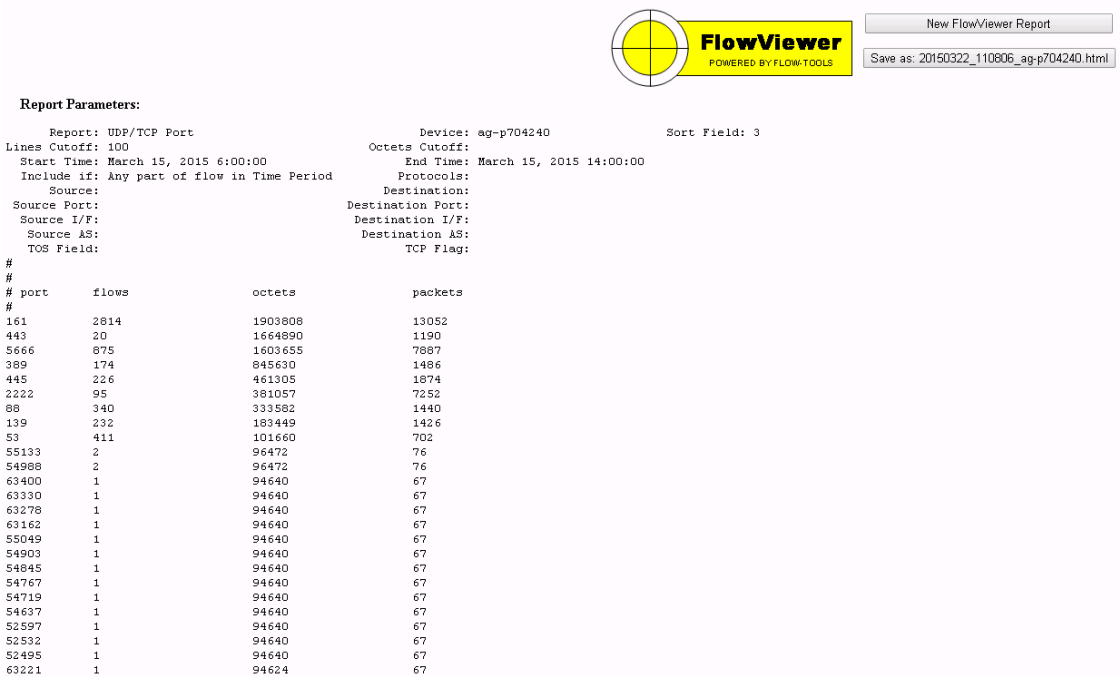
```
$prikaz = "ps -ef | grep flow";

exec("ssh -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -
    i ".$ssh_klic." ".$ssh_uzivatel."@".$ip_serveru." '".$prikaz."'",$datasrv);

for ($x=0;$x<count($datasrv);$x++){
    echo $datasrv[$x]."<br/>";
    // Zjisteni , zda bezi na serveru FLOW
    $find_proces = strstr($datasrv[$x], $dns_router_local);
    if ($find_proces != "") {
    $server_flow = "UP";
  }
}

// Flow je zapnute i na servereru i na routeru presmerovavam na FLOW, jinak
    vypisuji info, ze Flow neni zaple
if ( $server_flow == "UP" AND $router_flow == "UP" ){
  echo '<meta http-equiv="refresh" content="0;url=http://1.1.1.1/FlowViewer/
      index2.php?dns_router_local='.$dns_router_local.'">';
}else {
```

```
    echo '<meta http-equiv="refresh" content="0;url=http://2.2.2.2/dohledkcp/
        flowoff.php">';
}
```

Výpis 4: Volání sytémového příkazu z PHP



Obrázek 15: Ukázka výpisu aplikace NetFlow



FlowViewer
POWERED BY FLOW-TOOLS

New FlowViewer Report

Save as: 20150322_110806_ag-p704240.html

Report Parameters:

```
      Report: UDP/TCP Port                          Device: ag-p704240              Sort Field: 3
Lines Cutoff: 100                                Octets Cutoff:
  Start Time: March 15, 2015 6:00:00                 End Time: March 15, 2015 14:00:00
  Include if: Any part of flow in Time Period      Protocols:
      Source:                                    Destination:
 Source Port:                                Destination Port:
  Source I/F:                                 Destination I/F:
   Source AS:                                  Destination AS:
   TOS Field:                                        TCP Flag:
#
#
# port      flows                octets               packets
#
161         2814                 1903808              13052
443         20                   1664890              1190
5666        875                  1603655              7887
389         174                  845630               1486
445         226                  461305               1874
2222        95                   381057               7252
88          340                  333582               1440
139         232                  183449               1426
53          411                  101660               702
55133       2                    96472                76
54988       2                    96472                76
63400       1                    94640                67
63330       1                    94640                67
63278       1                    94640                67
63162       1                    94640                67
55049       1                    94640                67
54903       1                    94640                67
54845       1                    94640                67
54767       1                    94640                67
54719       1                    94640                67
54637       1                    94640                67
52597       1                    94640                67
52532       1                    94640                67
52495       1                    94640                67
63221       1                    94624                67
```

Obrázek 16: Pohled na vytížení síťových portů

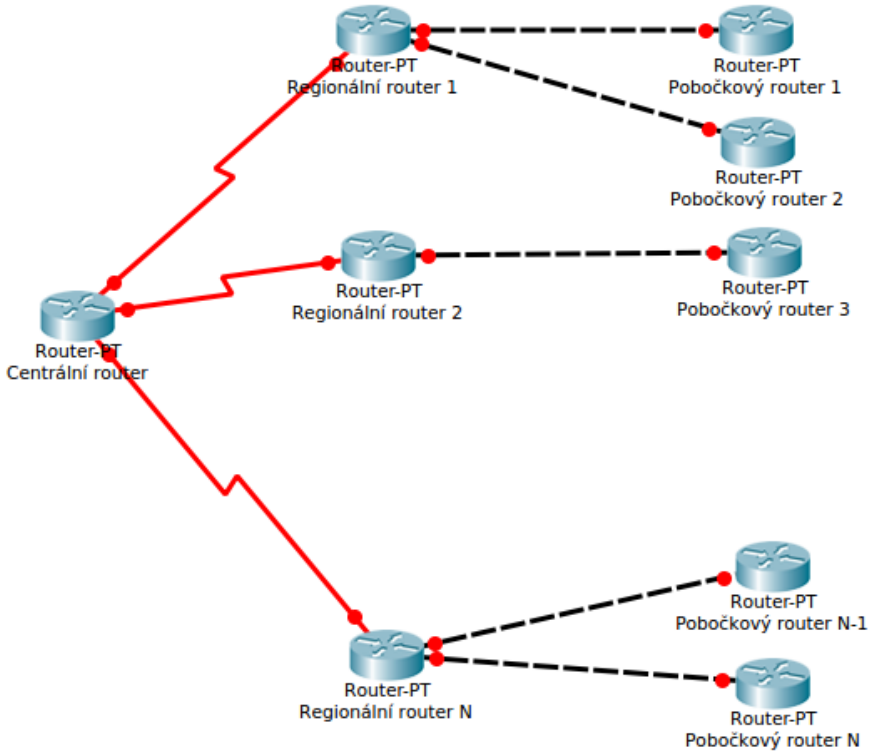
5 Generování reportů

V každé firmě čas od času nastává situace, kdy je vyžadován nějaký typ reportu, ať už pro potřeby vlastního přehledu nebo prezentaci dat. V tomto případě se budeme bavit o reportu všech aktivních prvků, které jsou v době generování reportu nakonfigurovány a připojeny do sítě. Způsobů jak tohoto docílit je zajisté více. Otázkou zůstává, který je ten správný.

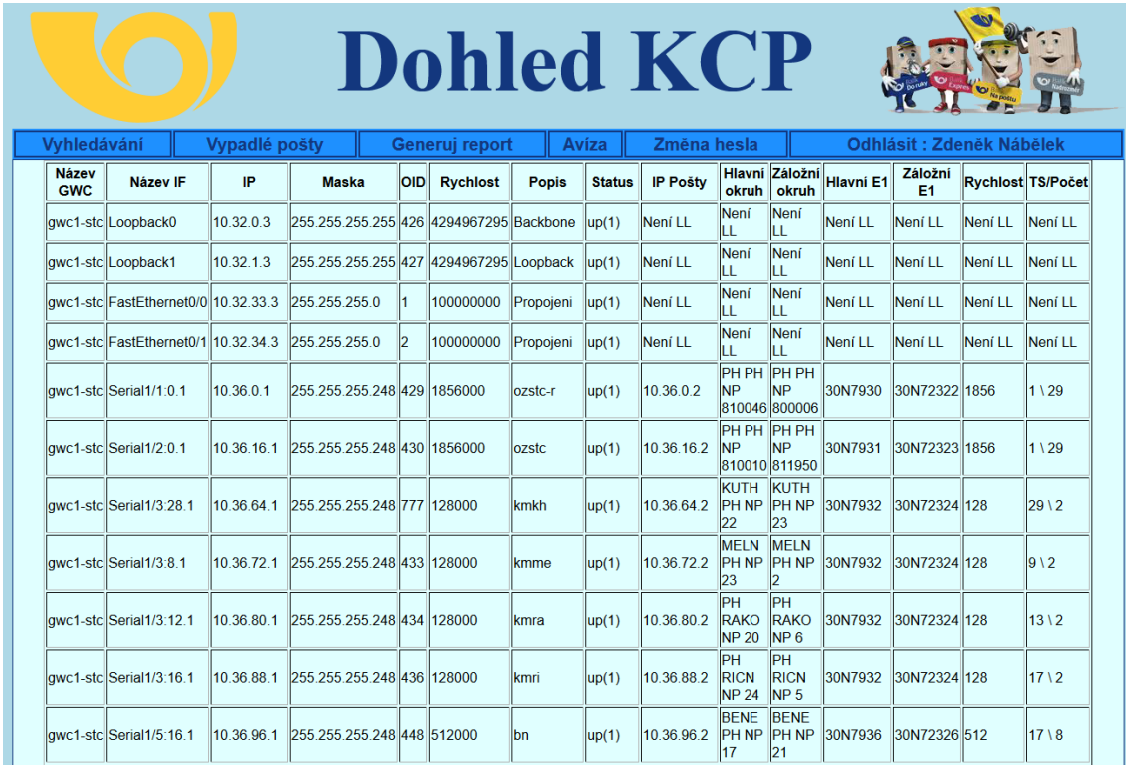
5.1 Online report

Vycházejme tedy z předpokladu, že je vyžadován vždy aktuální výstup. Získat tato data není problém prostřednictvím vhodně napsaného selektu nad databází. Co se však stane v případě, že se zapomněly nějaké nové prvky přidat, vyřazené prvky vymazat nebo stávající prvky aktualizovat. Pak by poslední získaná data z databáze úplně neodpovídala skutečnosti.

Proto byl zvolen následující alternativní přístup k dosažení aktuálního výstupu. Report se generuje na základě reálné konfigurace a získaných dat z jednotlivých zařízení. Aplikace vychází ze všeobecné topologie (obr. 17), ve které postupně prochází aktivní síťové prvky a tak získává informace až ke koncovým routerům. Tyto informace se zpracovávají a postupně ukládají do textového souboru. Obdržený soubor je pak pomocí dalšího skriptu načten a získaná data jsou zpracována až ke koncovým zařízením. Výsledná data jsou pracovníkovi prezentována v přehledné tabulce, kterou si může stáhnout nebo vytisknout (obr. 18), [14].



Obrázek 17: Všeobecné schéma síťové topologie



Dohled KCP

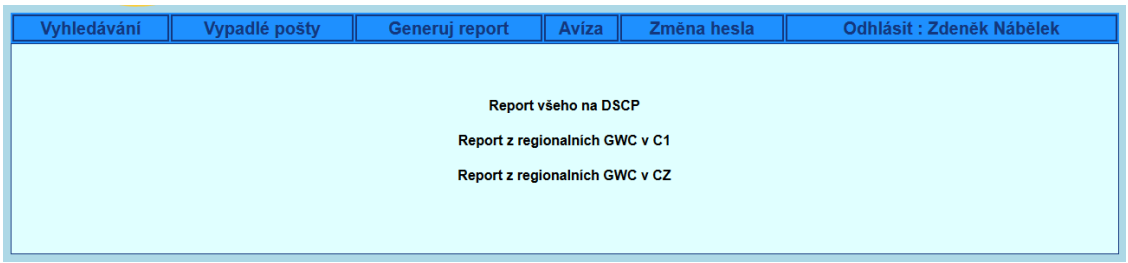
Vyhledávání | Vypadlé pošty | Generuj report | Avíza | Změna hesla | Odhlásit : Zdeněk Nábělek

Název GWC	Název IF	IP	Maska	OID	Rychlost	Popis	Status	IP Pošty	Hlavní okruh	Záložní okruh	Hlavní E1	Záložní E1	Rychlost	TS/Počet
gwc1-stc	Loopback0	10.32.0.3	255.255.255.255	426	4294967295	Backbone	up(1)	Není LL	Není LL	Není LL	Není LL	Není LL	Není LL	Není LL
gwc1-stc	Loopback1	10.32.1.3	255.255.255.255	427	4294967295	Loopback	up(1)	Není LL	Není LL	Není LL	Není LL	Není LL	Není LL	Není LL
gwc1-stc	FastEthernet0/0	10.32.33.3	255.255.255.0	1	100000000	Propojeni	up(1)	Není LL	Není LL	Není LL	Není LL	Není LL	Není LL	Není LL
gwc1-stc	FastEthernet0/1	10.32.34.3	255.255.255.0	2	100000000	Propojeni	up(1)	Není LL	Není LL	Není LL	Není LL	Není LL	Není LL	Není LL
gwc1-stc	Serial1/1:0.1	10.36.0.1	255.255.255.248	429	1856000	ozstc-r	up(1)	10.36.0.2	PH PH NP 810046	PH PH NP 800006	30N7930	30N72322	1856	1 \ 29
gwc1-stc	Serial1/2:0.1	10.36.16.1	255.255.255.248	430	1856000	ozstc	up(1)	10.36.16.2	PH PH NP 810010	PH PH NP 811950	30N7931	30N72323	1856	1 \ 29
gwc1-stc	Serial1/3:28.1	10.36.64.1	255.255.255.248	777	128000	kmkh	up(1)	10.36.64.2	KUTH PH NP 22	KUTH PH NP 23	30N7932	30N72324	128	29 \ 2
gwc1-stc	Serial1/3:8.1	10.36.72.1	255.255.255.248	433	128000	kmme	up(1)	10.36.72.2	MELN PH NP 23	MELN PH NP 2	30N7932	30N72324	128	9 \ 2
gwc1-stc	Serial1/3:12.1	10.36.80.1	255.255.255.248	434	128000	kmra	up(1)	10.36.80.2	PH RAKO NP 20	PH RAKO NP 6	30N7932	30N72324	128	13 \ 2
gwc1-stc	Serial1/3:16.1	10.36.88.1	255.255.255.248	436	128000	kmri	up(1)	10.36.88.2	PH RICN NP 24	PH RICN NP 5	30N7932	30N72324	128	17 \ 2
gwc1-stc	Serial1/5:16.1	10.36.96.1	255.255.255.248	448	512000	bn	up(1)	10.36.96.2	BENE PH NP 17	BENE PH NP 21	30N7936	30N72326	512	17 \ 8

Obrázek 18: Ukázka výpisu získaných dat uživateli

5.2 Výpis obsahu databáze

Samozřejmě ale na výpis dat přímo z databáze nemůžeme úplně zapomenout. Takže tato funkcionalita byla implementována a pracovník dohledového centra si tak může sám zvolit, o který typ výpisu má zájem (obr. 19), [4].



Obrázek 19: Uživatelské rozhodnutí o jaký typ reportu půjde

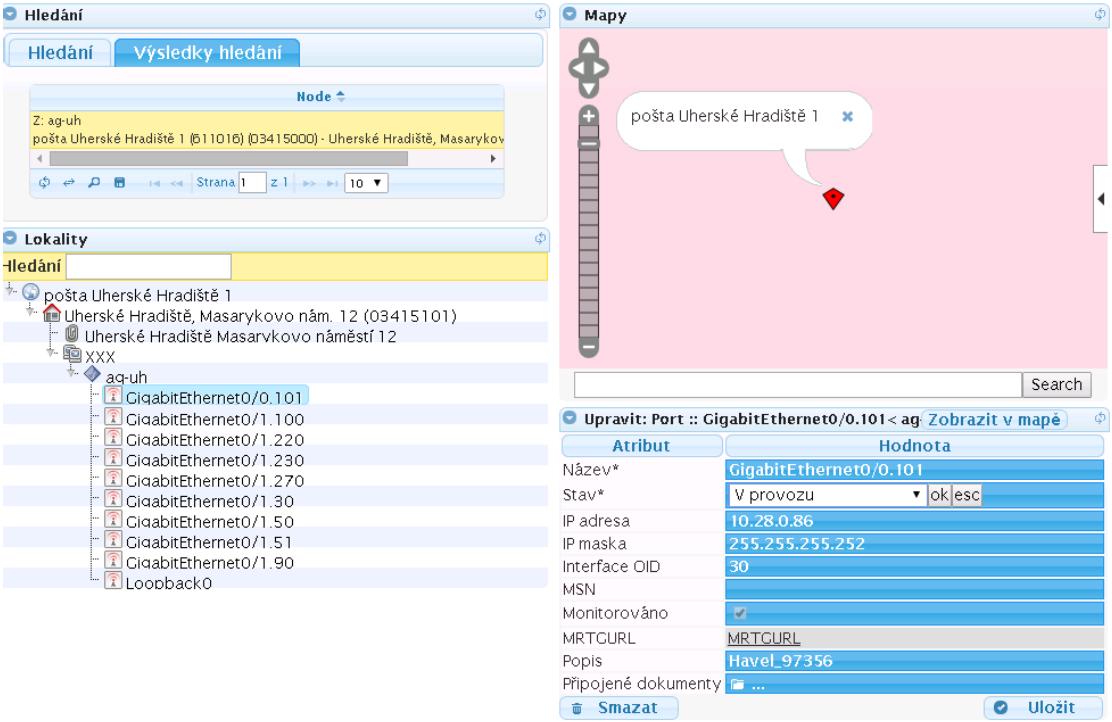
Po zvolení reportu z databáze je vygenerován soubor, jehož jedinečný název tvoří vždy aktuální datum. Informace o bližším postupu a odkaz pro stáhnutí souboru je následně prezentován uživateli (obr. 20).

Obrázek 20: Celkové informace o získaných datech

6 Úprava a generování podkladů pro NI

Můžeme mít důmyslně propracovaný systém, ale nevyhnutelně jednou přijde čas, kdy bude zapotřebí nějakým způsobem upravit evidované zařízení. Pokud toto řešíme na malé síti, kde jsou řádově jednotky těchto prvků, pak jistě není problém tuto změnu provést ručně. Uvažujme ale situaci, že síťová topologie je tvořena 4000 prvky. Jelikož každý prvek má v databázi uloženo deset portů a každý port nese šest různých údajů, je výsledný počet údajů vysoké číslo. Nastává tedy otázka, zda lze tento proces nějak automatizovat. Odpověď zní ano.

Jak již bylo dříve uvedeno, pro evidenci síťových zařízení se využívá programu NI. Protože je zajištěna distribuce údajů z tohoto systémů do dalších dvou databází a tří programů, musí se počítat s propagací změn a synchronizací mezi databázemi. Synchronizace je realizována jedenkrát za hodinu. Pro ilustraci je uveden jeden konkrétní případ zařízení a jeho portů (obr. 21).



Obrázek 21: Ukázka výpisu síťového zařízení z NI

6.1 XML

Extensible Markup Language, zkráceně XML, popisuje třídu datových objektů nazývaných XML dokumenty a částečně popisuje chování počítačových programů, které je zpracovávají. XML je profil aplikace nebo omezená forma SGML. Podle konstrukce XML dokumenty jsou v souladu SGML dokumenty. XML dokumenty jsou tvořeny z paměťových jednotek nazývaných entity, které obsahují buď analyzované nebo neanalyzované údaje. Analyzované údaje se skládají ze znaků, z nichž některé tvoří znaková data a některé z nich tvoří značky. Značky kódují popis uložení rozložení a logické struktury dokumentu. XML poskytuje mechanismus pro zavedení omezení úložiště rozložení a logické struktury.

Při zakládání či úpravě síťových zařízení by se uživatel neměl zdržovat přihlašováním do různých systémů a vyplňováním několika formulářů. K zefektivnění této činnosti byla do diplomové práce doplněna funkcionalita automatického generování XML souboru, který lze pak jednoduše naimportovat do systému NI.

Samotný XML soubor, který bude použit pro import, má klasickou strukturu a celý jej naleznete v příloze. Zde je k dispozici jen jeho začátek (výpis 5).

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>2.0</version>
    <date>2015-03-20T11:00:00Z</date>
    <hosts>
        <host>
            <host>ag-p123456</host>
            <name>ag-p123456</name>
            <proxy/>
            <status>0</status>
            <ipmi_authtype>-1</ipmi_authtype>
            <ipmi_privilege>2</ipmi_privilege>
            <ipmi_username/>
            <ipmi_password/>
            <templates>
                <template>
                    <name>CpTemplateOffice</name>
                </template>
            </templates>
            <groups>
                <group>
                    <name>POSTOFFICE</name>
                </group>
                <group>
                    <name>R_VC</name>
```

```
    </group>
    <group>
        <name>T_NT</name>
    </group>
</groups>
<interfaces>
    <interface>
        <default>1</default>
        <type>1</type>
        <useip>1</useip>
        <ip>10.10.10.10</ip>
        <dns/>
        <port>10050</port>
        <interface_ref>if1</interface_ref>
    </interface>
...
vzhledem k obsáhlosti tohoto souboru byl výpis zkrácen.
```

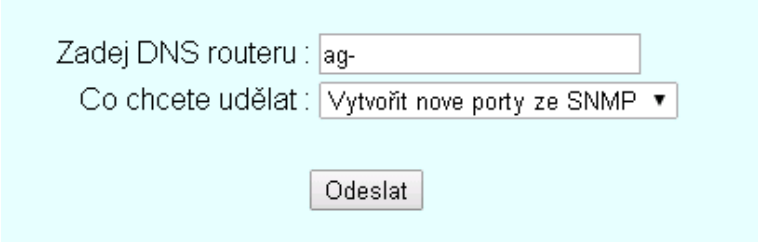
Výpis 5: Struktura vygenerovaného XML souboru

6.2 Změna portů síťového zařízení

Pokud budeme chtít stávající porty změnit nebo přidat nové, musíme dodržovat postup daný výrobcem software:

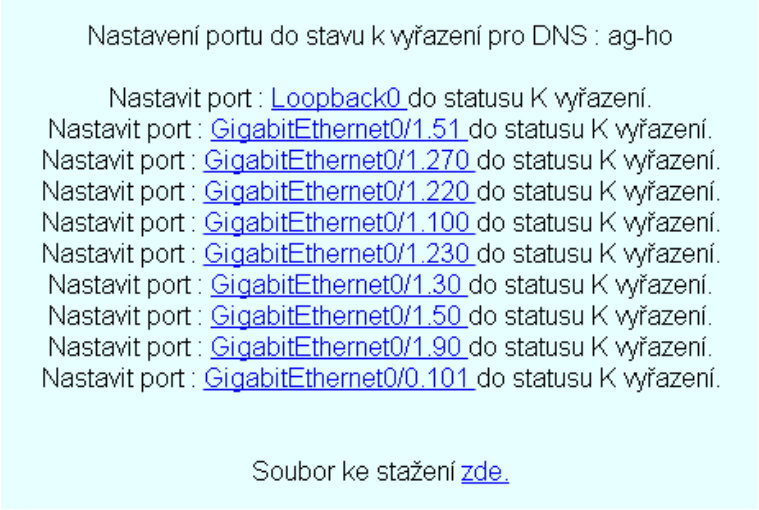
- Nastavení portu do stavu k vyřazení.
- Prodleva k nutné synchronizaci mezi DB.
- Smazání starých portů a jejich údajů.
- Založení nových portů a vyplnění potřebných informací.

Tento proces je opravdu zdlouhavý a náročný na mechanickou práci. Příkladem může být obnova síťových zařízení nebo jen změna typu připojení a změna konfigurace routeru. V plošném měřítku to jsou desítky možná stovky hodin ruční práce. Denně se zhruba takto aktualizuje pět až deset síťových zařízení. Proto byla zvolena metoda, která bude programu "podsouvat"skriptem vyplněný formulář a vše se tak bude automatizovat. Jediné, co se bude po uživateli vyžadovat, je aby vyplnil (obr. 22) konkrétní síťové zařízení, o které se mu jedná, a způsob jakým s ním bude nadále pracovat.



Obrázek 22: Požadavek na zadání zařízení a zvolení operace

Je-li dopředu znám princip a metodika postupu, mohou být některé kroky trochu zjednodušeny. Například, pokud je potřeba smazat konkrétní port ze zařízení, nelze se kvůli replikaci dat vyhnout mezikroku změny statusu. Při změně statusu na stav "K vyřazení"(obr. 23) je možné část údajů smazat a ponechat jen ty, které jsou důležité pro synchronizaci. Ruční práce uživatele je nahrazena dotazem do databáze, ze které se získají potřebné údaje. Aplikace připraví odkaz, který supluje odeslání formuláře (výpis 6). Popsaný proces aktualizace údajů se nazývá update a ve skriptu je značen písmenem "u", [9, 11].



Obrázek 23: Výsledná obrazovka pro aktualizaci stavu portu

```
https://ni.cpost.cz/loc/locEditSave.htm?parent=1234567&name=Serial0/3/0.1&status
    =8&objId=9876543&objType=S&search=false
```

Výpis 6: Sestavení odkazu pro aktualizaci portu

Analogicky postupujeme i při dokončení mazání portů. Rozdíl je nyní jen v jiném typu sestaveného odkazu (výpis 7).

```
https://ni.cpost.cz/loc/locEditSave.htm?objId=9876543&id=9876543&objType=S&type=
    S&delete=true&parentId=1234567&parent=1234567
```

Výpis 7: Sestavení odkazu pro smazání portu z databáze

6.3 Online výčet konfigurace a založení portu

Trochu složitější situace nastává, chceme-li založit nové porty. Vytvoření nových portů nelze provést současně se změnou jejich statusu "K vyřazení"skrze danou funkcionalitu aplikace NI, která udává hodinový interval na synchronizaci mezi jednotlivými databázemi a zabraňuje tak duplicitě názvů portů. Je potřeba si uvědomit, že pro založení portů na konkrétním zařízení, je nutno znát několik hodnot:

- Název rozhraní.
- IP adresu.
- Použitou sítovou masku.
- Status.
- OID.
- Příznak monitorování.
- Popis.

Tyto údaje však nikde v databázi nenajedeme, protože tam zatím nebyly nevloženy. Jak tedy potřebné informace získat? Je možné zvolit některou ze základních metod:

- Ruční přihlášení na router.
- Využití scriptu prezentace dat z routeru.
- Pomocí protokolu SNMP.

Cílem je především proces zakládání portů zcela automatizovat a vyloučit lidský faktor. Zbývají tedy pouze dvě možnosti. Prezentace dat z routeru nebude v tomto případě vhodná, protože procházet získaný výstup by bylo složité. Daleko jednodušší bude použít metodu SNMP. Jsou tedy k dispozici všechny potřebné parametry a může se opět přejít k sestavení odkazu simulujícího vyplnění a odeslání formuláře (výpis 8).

```
https://ni.cpost.cz/loc/locEditSave.htm?parent=1234567&name=GigabitEthernet0
    /0.101&status=5&cpIpAddress=10.10.10.10&cpIpMask=255.255.255.252&
    cpInterface=40&cpMsn=&cpMonitored=true&cpComment=WAN&objId=0&objType=S
    &search=falsee
```

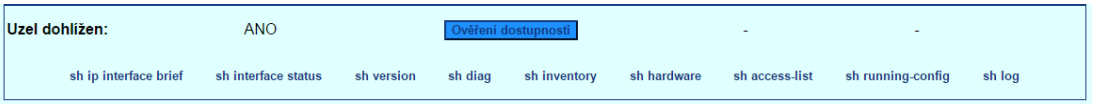
Výpis 8: Sestavení odkazu pro vytvoření nového portu

7 Prezentace dat z routeru

Dohledový systém byl navržen tak, aby umožnil jeho uživateli online práci se síťovým zařízením. Z několikaměsíčního pozorování bylo zjištěno, že nejčastějším typem příkazu, který je pracovníky zadáván pro získání potřebných informací z routeru, je typ show (na routeru používáno zkráceně jako sh). Pro názornost je na obr. 24 ukázán výstup příkazu "sh ip interface brief". Ty nejvíce používané příkazy byly implementovány do rozhraní aplikace (obr. 25), [1, 2].

```
Pripojuji se telnetem na 10.60.144.2...Connected1

Zacatek vystupu
--------------------------------------------------------------------------------------------------------------------------
ag-ov#sh ip interface brief
Interface                  IP-Address      OK?  Method  Status                Protocol
GigabitEthernet0/0         10.120.129.1    YES  NVRAM   down                  down
GigabitEthernet0/1         unassigned      YES  NVRAM   up                    up
GigabitEthernet0/1.10      10.120.130.1    YES  NVRAM   up                    up
GigabitEthernet0/1.20      10.120.8.1      YES  NVRAM   up                    up
GigabitEthernet0/1.30      10.120.9.65     YES  NVRAM   up                    up
GigabitEthernet0/1.31      10.120.9.129    YES  NVRAM   up                    up
GigabitEthernet0/1.40      10.158.73.65    YES  NVRAM   up                    up
GigabitEthernet0/1.50      10.157.10.65    YES  NVRAM   up                    up
GigabitEthernet0/1.500     10.120.11.1     YES  NVRAM   up                    up
ATM0/1/0                   unassigned      YES  NVRAM   administratively down down
BRI0/1/0                   10.60.144.10    YES  NVRAM   up                    up
BRI0/1/0:1                 unassigned      YES  unset   down                  down
BRI0/1/0:2                 unassigned      YES  unset   down                  down
Serial0/2/0                unassigned      YES  NVRAM   up                    up
Serial0/2/0.1              10.60.144.2     YES  NVRAM   up                    up
GigabitEthernet1/0         10.60.144.17    YES  NVRAM   up                    up
IDS-Sensor3/0              unassigned      YES  NVRAM   administratively down down
Loopback0                  10.60.144.254   YES  NVRAM   up                    up
Loopback1                  10.120.128.1    YES  NVRAM   up                    up
--------------------------------------------------------------------------------------------------------------------------
Konec vystupu:
```

Obrázek 24: Online výpis IF na routeru



Obrázek 25: Umístění příkazů v aplikaci

Po kliknutí na vybraný příkaz se otevře nová webová stránka, která volá funkci k provedení vzdáleného příkazu (výpis 9) a která zobrazí aktuální výstup. Stejný výstup by pracovník viděl po zadání vybraného příkazu na konkrétním prvku, kde je přihlášen. Výhody tohoto přístupu spočívají ve značné časové úspoře, dále pak odpadá nutnost školit zaměstnance pro práci s příkazovým řádkem síťových zařízení a také si pracovník nemusí pamatovat příkazy. Vzhledem k tomu, že je ve skriptu používán jednotný přístup, nemusí se každému uživateli zvlášť generovat přihlašovací údaje pro přístup na síťová zařízení, [15].

```
function GetOutputOf($_command){
    if (!$this->connected) return false;

    $this->SendString($_command);

    $output = array();
    $work = true;

    while( $work && $data = $this->ReadTo( array("--More--","#") ) ){
      $null_data = true;
      foreach($data as $line){
        if (trim($line) != "") {$null_data = false;break;}
      }
      if ($null_data) { break;}

      if ( trim($data[count($data)-1]) == '--More--'){
        unset($data[count($data)-1]);

        if ( trim($data[1]) == '' ) unset($data[1]);
        if ( strpos($data[0], $_command)!==FALSE ) unset($data[0]);

        fputs($this->socket, " ");
      }

      if ( strpos($data[count($data)-1], '#')!==FALSE /* || (count($data) == 1 && $data
          [0] == "")*/ ){
        unset($data[count($data)-1]);

        if ( trim($data[1]) == '' ) unset($data[1]);
        if ( strpos($data[0], $_command)!==FALSE ) unset($data[0]);
        $work = false;
      }

      for($i = count($data)-1; $i>0; $i--){
        if (trim($data[$i]) == "") unset($data[$i]);
        else break;
      }

      foreach($data as $v){
         $output[] = $v;
      }
    }
    return $output;
}
```

Výpis 9: Vzdálené volaní příkazu na routeru

8 Reálný test dostupnosti a výčet parametru IF

Na dohledovém centru musí být pracovníci schopni podávat aktuální informace po telefonu a provádět reálné testy v provozu. Tedy je zapotřebí zjistit dostupnost zařízení nebo konfiguraci jednotlivých síťových prvků. Co se týká dostupnosti, zde mohou dle typu konektivity nastat dvě situace.

V běžných situacích, se kterými se setkáváme, je většinou zařízení připojeno do sítě jedním rozhraním. Nejčastěji to bývá ethernet. Přes tento typ rozhraní bývají nejčastěji připojeny IPSec tunely, IP Connecet, různé typy DSL nebo NT.

Speciálním typem jsou tzv. LL okruhy připojené přes sériové rozhraní. Zde se používají linky firmy O2, které jsou zakončeny v centru a na pobočce. V centrech se používá technoligoe E1 složená z několika TS. Dle rychlosti jsou postupně obsazovány jednotlivé TS a konfigurovány pro konkrétní uzel. Na pobočce je linka zakončena nejčastěji modemem a k routeru je připojena pomocí sériového rozhraní. K tomuto typu konektivity je zřízená i záložní linka - ISDN, která je zakončena na BRI rozhraní a v případě výpadku hlavní linky se stává aktivní. V detailním výpise každého uzlu je zobrazen typ konektivity, výpis zakončení v centru a na uzlu.

8.1 ICMP dostupnost

Test dostupnosti se provádí dle typu přípojky na uzlu. Jak pracovník tento test provádí je ukázáno na obr. 26. Z rozhraní programu je zavolán systémový skript, který podle předané IP adresy provede ping. Výsledky pingu jsou zpracovány a na základě úspěšnosti vyhodnoceny.

Obrázek 26: Test dostupnosti zařízení

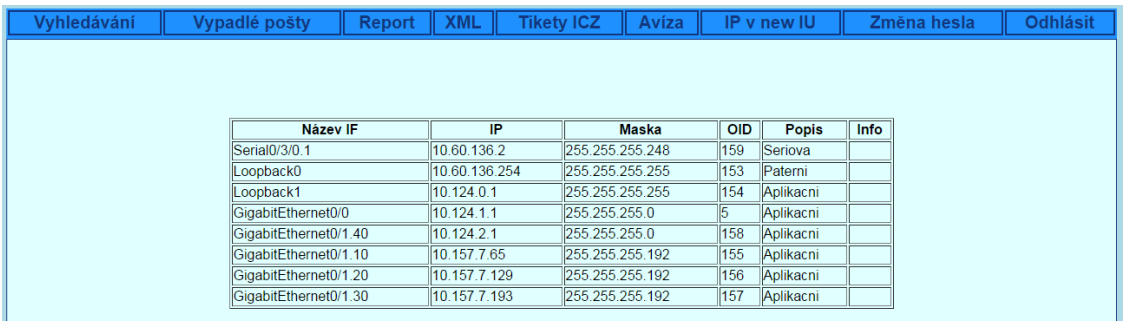
8.2 SNMP výčet

Výčet informací pomocí protokolu SNMP je v programu využíván na více místech. Slouží k tomu, aby byly zobrazované informace co nejpřesnější a v případě potřeby měl pracovník možnost porovnat hodnoty z databáze vůči reálným hod-

notám z routeru. Takovým konkrétním případem je například výčet portů a jejich vlastností. V databázi je uložena hodnota, která byla získaná při zřizování síťového prvku. Občas ale dojde k rozšíření virtuální IF na routeru nebo doplnění fyzických portů. Potom se na první pohled může zdát, že přibyl jen další port, ale situace je trochu složitější. Není-li na routeru nastaveni persistentní mód (jeho konfigurace viz výpis 10), tak při přidání pouze virtuálního IF dojde k reindexaci OID. Díky tomu se může stát, že je zapnuto měření konkrétního rozhraní na základě známého OID. Po reindexaci OID měření v tom lepším případě přestane fungovat. V opačném případě bude měření pokračovat, ale na jiném IF než jsme chtěli. Z tohoto důvodu byl vytvořen aktivní odkaz na propojovací IP adrese. Po jeho rozkliknutí se zavolá skript, díky kterému se ze zařízení vyčtou všechny potřebné parametry a pracovník je ihned schopen porovnat, zda je vše v pořádku a prezentovaná data mají vypovídající charakter (obr. 27).

```
Router(config)# snmp-server ifindex persist
Router(config-if)# snmp-server ifindex persist
```

Výpis 10: Nastavení persistentního modu na routeru



Název IF	IP	Maska	OID	Popis	Info
Serial0/3/0.1	10.60.136.2	255.255.255.248	159	Seriova	
Loopback0	10.60.136.254	255.255.255.255	153	Paterni	
Loopback1	10.124.0.1	255.255.255.255	154	Aplikacni	
GigabitEthernet0/0	10.124.1.1	255.255.255.0	5	Aplikacni	
GigabitEthernet0/1.40	10.124.2.1	255.255.255.0	158	Aplikacni	
GigabitEthernet0/1.10	10.157.7.65	255.255.255.192	155	Aplikacni	
GigabitEthernet0/1.20	10.157.7.129	255.255.255.192	156	Aplikacni	
GigabitEthernet0/1.30	10.157.7.193	255.255.255.192	157	Aplikacni	

Obrázek 27: Online získání informaci o IF z routeru

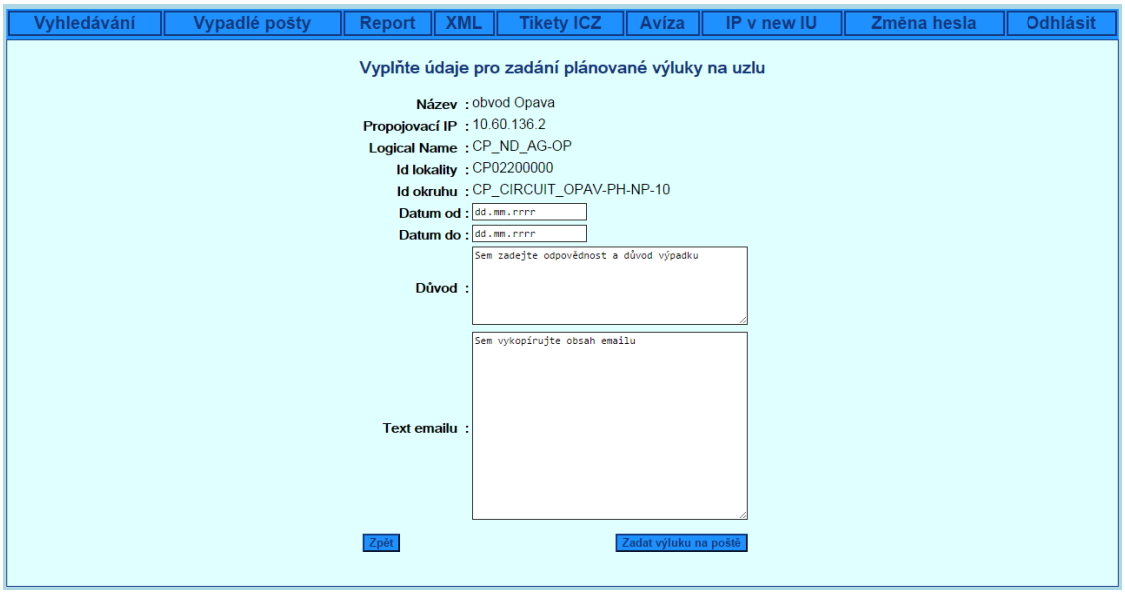
9 Zadávání a přehled výluk

Pro potřeby běžného provozu bylo v aplikaci vytvořeno rozhraní, kde mohou zaměstnanci zadávat výluky na jednotlivých uzlech. Je to z důvodu, že jsou ze stran poskytovatelů konektivity plánovány opravy, reinstalace, výměny zařízení či servisní zásahy, které mají za následek nedostupnost koncových zařízení. V databázi je vytvořena tabulka (výpis 11), do které se evidují základní informace jako zodpovědnost za výpadek, datum, čas, popis nebo text operátora, který jej do systému zadal, [6, 7].

```
CREATE TABLE dohled_kcp.outageIs (
  idOutage INT(6) NOT NULL AUTO_INCREMENT,
  name VARCHAR(50) NOT NULL,
  cpNode VARCHAR(50) NOT NULL,
  cpObject VARCHAR(50) NOT NULL,
  wanIP VARCHAR(50) NOT NULL,
  dateFrom VARCHAR(20) NOT NULL,
  dateTo VARCHAR(20) NOT NULL,
  firm  VARCHAR(500) NOT NULL,
  logicalName VARCHAR(50) NOT NULL,
  location  VARCHAR(50) NOT NULL,
  idCircuitOutage VARCHAR(100) NOT NULL,
  dateWrite VARCHAR(30) NOT NULL,
  user VARCHAR(100) NOT NULL,
  email LONGTEXT NOT NULL,
  PRIMARY KEY (idOutage)
)
```

Výpis 11: Založení databáze pro výluky

Jakmile jsou údaje vloženy do formuláře (obr. 28), formulář se odešle a zadaná data se vloží do databáze. Současně se provede odeslání emailu osobám, které mají být informovány o těchto výlukách (výpis 12).



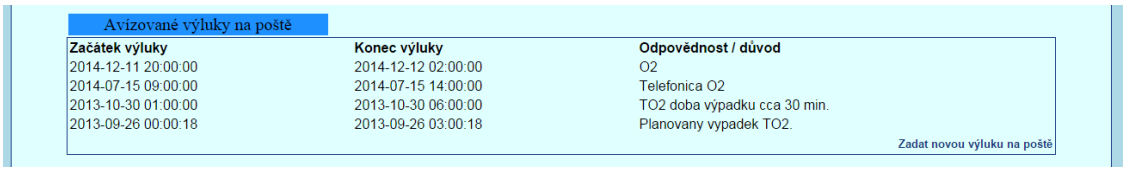
Obrázek 28: Zadávání výluky zaměstnancem

```
if (ISSET($_POST["set"])) {

    $dateFrom = $_POST["dateFrom"];
    $dateTo = $_POST["dateTo"];
    $firm  = $_POST["firm"];
    $email = $_POST['email'];
    $logicalName = $_POST['setLogicalName'];
    $location  = $_POST['setLocation'];
    $wanIP = $_POST['setWanIP'];
    $name = $_POST['setName'];
    $idCircuitOutage = $_POST['setidCircuit' ];
    $dns = $_POST['setDns'];
    $rcp = "vypis emailovych adres";
    $headers = "Content-type: text/plain; charset=utf-8\n";
    $headers .= "From:avizovane vyluky";
    $subject = "Avizovana vyluka na DSCP";
    $emailMessage = "Pošta: $name\n\n"
            . "Odkaz: \thttp://dohled.cpost.cz/nodesout.php\n\n"
            . "Od: \t$dateFrom\n"
            . "Do: \t$dateTo\n\n"
            . "Dns: \t$dns\n"
            . "WanIP: $wanIP\n"
            . "Okruh: \t$idCircuitOutage\n\n"
            . "Odpovědnost: \n$firm\n\n"
            . "Obsah emailu: \n$email";

    $outage->setOutageIs($logicalName, $wanIP, $location, $name, $dateFrom,
        $dateTo, $firm, $idCircuitOutage, $email);
```

```
        mail($rcp, $subject, $emailMessage, $headers);
        $redir->makeRedirectWithTwoParam('vysledek.php', 'loc', $location, 'dns', $dns);
    }
```

Výpis 12: Zpracování odeslaného formuláře

U každého uzlu jsou takto zadané výluky zobrazeny v detailním výpise. Aby se nehromadilo zbytečně moc informací, je zobrazeno vždy jen posledních pět záznamů (obr. 29).

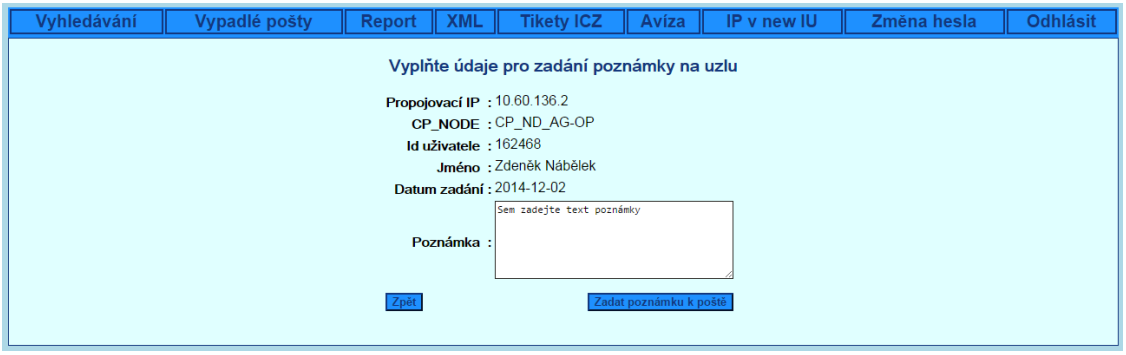


Obrázek 29: Zobrazení výluk na konkrétním uzlu

S každým výpadkem, který je zaznamenán, se provede kontrola, zda se vypadlý uzel nachází v tabulce výluk. Pokud ano, jsou tyto informace načteny a zobrazeny přímo na přehledu vypadlých uzlů (kapitola 11).

10 Možnost předávání informací

Ve většině případu je v dohledových centrech nepřetržitý provoz. Ať už je provoz dvou nebo tří směnný, zaměstnanci se střídají a předávají si informace. Některé jsou aktuální jen na pár hodin. Jiné na několik dnů či týdnů. Často tak dochází k nepřesnostem při předávání nebo časem k zapomenutí. Proto byla část aplikace vyčleněna této každodenní činnosti. Ke každému uzlu lze zapsat jakoukoli informaci (obr. 30). Kterýkoli zaměstnanec při otevření detailního výpisu ke konkrétnímu síťovému zařízení má možnost tuto informaci zobrazit nebo v případě její neaktuálnosti ji smazat (obr. 31). K evidenci této informace je opět využito databáze, tak jako je tomu v případě zadávání výluk. Vkládání dat je prováděno pomocí SQL dotazu typu INSERT (výpis 13), [13].



Obrázek 30: Předání informace - vytvoření

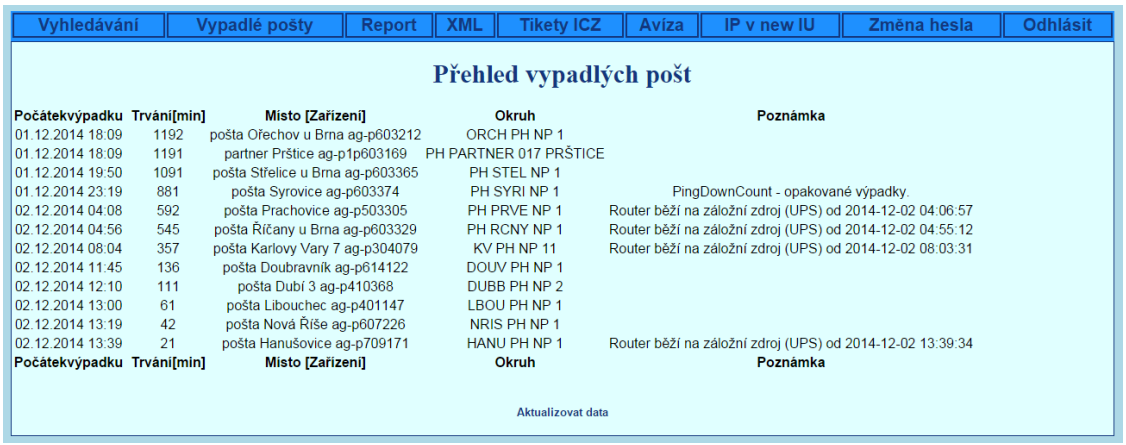
Obrázek 31: Předání informace - zobrazení

```
INSERT INTO surveillanceInfo(logicalName, wanIP, idUser, nameUser, date, message)
VALUES ('$logicalName', '$wanIP', '$idUser', '$nameUser', '$date', '$message')
```

Výpis 13: SQL info

11 Přehled vypadlých uzlů

Pro kontrolu dostupnosti jednotlivých síťových prvků je využito programu Zabbix, v němž jsou nastaveny politiky pro kontrolu dostupnosti. Jakmile jsou porušeny, je vygenerován trap, který je zaslán na centrální server - TrapConsole, kde jsou pomocí skriptu tyto údaje zpracovány. Jak již bylo popsáno v kapitole 9, je při zpracování trapu kontrolováno, zda vypadlý uzel je obsažen v plánovaných výlukách. Dále je založen incident do Service desk manageru a zobrazen výpadek na panelu dohledu. V menu programu této aplikace je položka Vypadlé pošty (obr. 32), která je na tento systém napojena a prezentuje aktuálně vypadlé uzly. Obsluze tak stačí jedno rozhraní a má vše potřebné ihned k dispozici.



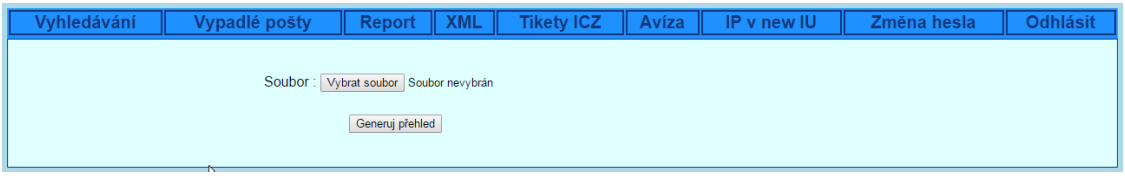
Obrázek 32: Přehled nedostupných uzlů

12 Počítání tiketů

V řadě případů firma využívá pomoci externích subjektů. Může tomu tak být z několika důvodů. Například nedostatečná kvalifikace zaměstnanců pro servisní úkony, nemožnost zajistit si vlastními prostředky nonstop pohotovost nebo jen proto, že tyto služby využívá zřídka a nevyplatí se takového zaměstnance "vychovat". Podíváme-li se na množství zařízení, které jsou na síti, zjistíme, že každý výrobce má jiný systém a mnohdy i jeden výrobce má specifické konfigurace pro své vlastní zařízení. Jeden z nejrozšířenějších výrobců síťových prvků na světe je společnost Cisco. Vyrábí switche, routery, F5, FW a mnoho dalších zařízení. Lidé, kteří zajišťují konfigurace a podporu, jsou certifikováni na jednotlivá zařízení. Jeden člověk tedy zajišťuje ASA FW, druhý routery a switche a další zase F5 a jiné balancery. Proto je při zadávání požadavků nutné rozlišovat, co kam patří. Pro potřeby reportů je v aplikaci vytvořeno rozhraní, které generuje celkové počty požadavků na jednotlivé externí pracovníky. Z pozorování provozu bylo zjištěno, že jednotlivé požadavky mohou být rozděleny do šesti základních skupin:

- ab_adv,
- ab_i,
- ab_iu,
- ab_flow,
- ab_f5,
- ostatní.

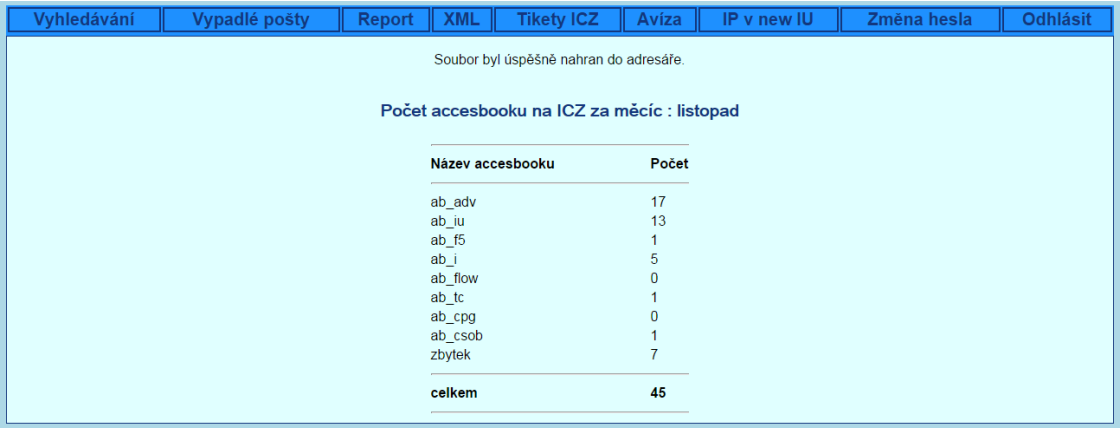
Názvy těchto skupin byly sestaveny ze zkratek ab, což je zkráceně accessbook, tedy konfigurační soubor obsahující pravidla pro povolovaní filtrace a specifikace jednotlivých komunikací. Druhá část názvu byla synteticky vytvořena a představuje skupinu prvků pro konfiguraci jako jsou routery, switche, FW, balancery atd. Toto značení se využívá ve spolupráci s externími firmami. Abychom byli schopni měsíčně poskytovat reporty tykající se počtu takto zadaných požadavků, vytvoříme k tomu část aplikace. V menu pro přepnutí do této části aplikace slouží položka Tikety. Objeví se formulář (obr. 33), který jako vstup očekává textový soubor, který se stáhne z rozhraní využívaného pro komunikaci s externí firmou. Jakmile se soubor nahraje, přichází na řadu script (jehož část vidíme ve výpise 14), který nám soubor rozebere a přesně spočítá kolik kterých požadavků bylo (obr. 34). Script současně využívá třídy classTicketCount.php, která zpracovává odeslaný soubor pomocí formuláře a pomocí systémových příkazů provádí výpočty. Ukázka výpočtu pro ab_adv je zobrazena ve výpise 15.



Obrázek 33: Načtení souboru se všemi tikety

```
if (ISSET($_POST["spocitej"])) {
    //nahrani souboru na disk
    $target_path = 'tikety/' . date('Ymd') . '.txt';
    if (move_uploaded_file($_FILES['uploadedfile']['tmp_name'], $target_path)) {
        echo "<br>Soubor byl úspěšně nahran do adresáře.<br><br>";
        $fileName = date('Ymd') . '.txt';
        include_once 'Class/classTicketCount.php';
        $ticket = new classTicketCount();
        $adv = $ticket->getNumberAbAdv($fileName);
        $iu = $ticket->getNumberAbIu($fileName);
        $f5 = $ticket->getNumberAbF5($fileName);
        $i = $ticket->getNumberAbI($fileName);
        $flow = $ticket->getNumberAbFlow($fileName);
        $tc = $ticket->getNumberAbTc($fileName);
        $cpg = $ticket->getNumberAbCpg($fileName);
        $csob = $ticket->getNumberAbCsob($fileName);
        $ziskanyPocet = $adv + $iu + $f5 + $i + $flow + $tc + $cpg + $csob;
        $zbytek = $ticket->getNumberZbytek($ziskanyPocet, $fileName);
        $celkem = $ticket->getNumberCelkem($fileName);
    } else {
        echo "Nastala chyba pri nahravani souboru!<br>";
    }
}
```

Výpis 14: Dělení a počítání externích tiketů dle skupin



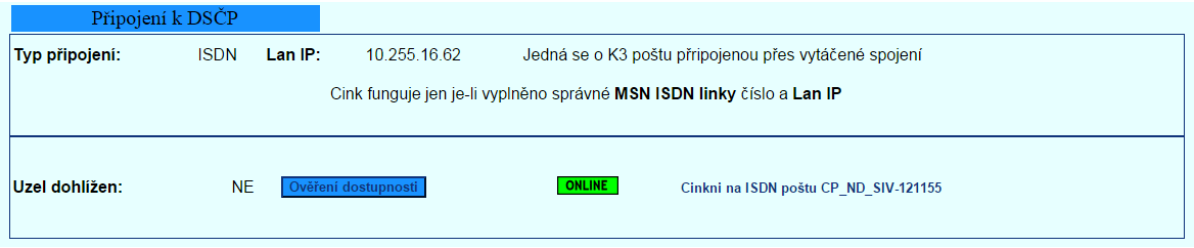
Obrázek 34: Rozbor a zobrazení přehledu tiketů

```
public function getNumberAbAdv($file) {
    exec('cat tikety/' . $file . ' | grep ab_adv -c', $output);
    return $output[0];
}
```

Výpis 15: Výpočet výskytu hledaného slova v dokumentu

13 ISDN

I v dnešní době, kdy není problém mít doma konektivitu do Internetu řádově v desítkách Mbit/s, stále existují i starší typu ISDN o rychlosti 64kbit/s. Tyto přípojky nejsou ale neustále spojeny. Do tzv. online režimu se dostávají až v případě potřeby komunikace. V praxi se ukázalo, že z cca 4000 přípojek je jich necelých 1700 tohoto typu. Nastává otázka jak docílit konektivity pro potřeby pracovníků dohledu, aby byly schopni otestovat, zda je pobočka v pořádku či nikoliv. Pro tyto účely byl vyvinut systémový script na linuxovém serveru vybavený modemem, který má za úkol cílovou stanici prozvonit. Na základě znalosti jejího telefonního čísla dojde k rozpoznání důvěryhodného zdroje a koncový uzel opětuje vytáčení, čímž se dostává do online režimu. Tento script avšak mohli obsluhovat pouze administrátoři tohoto serveru a nikoli pracovníci dohledu. Proto při identifikaci koncového uzlu typu ISDN byl na tlačítko "Ověření dostupnosti"(obr. 35) napojen php skript s názvem cinkni.php (výpis 17), který provede vzdálené přihlášení na server a tam provede spuštění systémového scritpu. Tímto způsobem může podle potřeby i pracovník dohledu provést spojení a ověřit potřebné údaje.



Obrázek 35: Test dostupnosti ISDN přípojek

14 Tvorba programu

Vzhledem k tomu, že výsledný program je nakonec uživateli prezentován formou www stránek, při jeho vytváření byl brán ohled na to, aby nebylo možné uložit si odkaz, který by mohl být zneužit neoprávněnými osobami. Z toho důvodu byla vytvořena třída CheckUser.php, která má za úkol ověřit, zda je uživatel přihlášen. Pokud tomu tak není, není uživateli povolena další práce s programem a uživatel je přesměrován na přihlašovací stránku.

Dále je třeba si uvědomit, že určité částí kódu se nám budou na stránkách opakovat. Proto je vhodné tyto části umístit do zvláštních souborů a pak provádět jen jejich vložení na potřebné místo. Typickým příkladem je začátek a konec souboru. V jednotlivých skriptech je takovéto volání zapsáno jako include_once 'Begin.php' respektive include_once 'End.php'. Jak již bylo v textu několikrát uvedeno, v programu je využito tříd. Díky vlastnostem objektového programování se mohou vytvořit objekty dané třídy a provádět tak volání přímo na ně. V takto vytvořeném programu se úpravy provedou pouze na jednom místě, ale projeví se všude tam, kde jsou tyto části využívány.

Pro práci s dočasnými informacemi se využívá session. Slouží nám pro uchování informací o přihlášeném uživateli a pro pomocné ukládání dat při získávaní hodnot pomocí SNMP apod. Jejich výhodou je, že ve standardním nastavení jsou aktivní jen po dobu 30 minut nebo do doby zavření prohlížeče. Tím docílíme toho, že neaktivní uživatel je z bezpečnostních důvodů v případě nečinnosti automaticky odhlášen. Tato doba se dá upravovat podle potřeb.

15 Testování aplikace v reálných podmínkách

Veškerá data a podklady byly shromažďovány z reálného provozu. Aplikace byla původně vyvíjena jako univerzální, ale s postupem času bylo zjištěno, že každý systém je natolik specifický, že vytvořit tento systém univerzální je téměř nemožné. Proto byly rozhraní a skripty uzpůsobeny přímo na míru potřebám pracovníků. Ve chvíli, kdy bylo doplněno vše, co vyvstalo z požadavků pracovníků, byla aplikace uvedena do provozu. Nejprve jen jako testovací systém, ve kterém se dolaďovaly chyby a prováděly kosmetické úpravy, později byl systém zaveden do ostrého provozu a dnes je již běžně využíván.

Testování aplikace provádělo devět pracovníků. Šest standardních uživatelů střídajících se na non-stop provozu a tři administrátoři prováděli všechny dostupné úkony jako je zakládání výluk, testování dostupnosti nebo generování reportu atd. Kontrola vložení a aktualizace dat byla prováděna v systémech, nad kterými tato aplikace pracuje.

16 Závěr

V rámci diplomové práce byla studována infrastruktura datové sítě České pošty. Dále byla prováděna analýza využívaných programů na dohledovém pracovišti a analýza uspořádání dat v několika databázích. Získané informace byly zpracovány a na jejich základě byla navržena struktura nové aplikace.

Prvotní tendence byly vytvořit program, který jen sjednotí práci nad několika systémy. S odstupem času se ale ukázalo, že to nebude dostačující. Pokud chceme, aby jakákoli aplikace byla využívána, musí splnit všechna očekávání a přinést něco navíc. Většinou to bude důvodem proč se začne využívat více než ty, které byly používaný doposud.

Každého asi zajímá, co nově vyvinutá aplikace pro Českou poštu přináší a jaké jsou její případné nevýhody. Nejprve se zaměříme na výhody této aplikace. Aplikace je rozdělena do několika sekcí. Každá sekce je složena z jednotlivých částí programového kódu. Vzhledem k těmto vlastnostem je aplikace velmi dobře rozšiřitelná. Jednoduchou úpravou je možné část kódu změnit nebo odmazat, aniž by to mělo vliv na chod programu. Aplikace má vlastní režim pro administrátory a klasické uživatele, což zaručuje, že konkrétní funkce budou dostupné jen pro konkrétní typ přihlášeného uživatele. Neaktivní uživatel je po určitém časovém úseku automaticky odhlášen a nemůže dojít k potenciálnímu zneužití této aplikace.

Výsledná aplikace má jedno rozhraní, díky kterému mají pracovníci dostupné veškeré informace a nemusí tak využívat více programů, aby dostali požadované výsledky. Dále aplikace umožňuje předávání informací o konkrétních síťových prvcích mezi pracovníky, zprostředkovává uživatelům informace a testy pro více než 3600 síťových prvků a nabízí zasílání emailových notifikací, například při založení výluky. Aplikace pracuje s pěti typy konektivity, umožňuje interaktivní práci s routery, obsahuje sedmdesát skriptů, z nichž je třicet tříd, komunikuje s třemi typy databází a také umí zpracovávat a generovat textové soubory. Bezpochyby jejím největším přínosem je zkrácení času potřebného pro získání požadovaných informací a eliminace lidského faktoru při zapisování informací do databází.

Vzhledem k tomu, že aplikace byla nasazena a zpřístupněná uživatelům od ledna tohoto roku, nebyly během takto krátké doby zjištěny žádné její nevýhody.

Při řešení diplomové práce jsem získal zkušenosti s návrhem komplexního systému pro dohledové středisko. Dále jsem byl obohacen o informace z oblasti programovacích jazyků, pochopení složitosti propojení mezi více programy a více typy databází.

Textová podoba diplomové práce byla sepsána pomocí šablony Diploma v prostředí LaTeX, [8].

17 Reference

[1] LAMMLE, Todd. *CCNA: výukový průvodce přípravou na zkoušku 640-802*. Vyd. 1. Brno: Computer Press, 2010, 928 s. ISBN 978-802-5123-591.

[2] EMPSON, Scott. *CCNA kompletní přehled příkazů: autorizovaný výukový průvodce*. Vyd. 1. Brno: Computer Press, 2009, 336 s. ISBN 978-80-251-2286-0.

[3] CROSS NETWORK INTELLIGENCE. *Cross-ni* [online]. 2013 [cit. 2015-04-19]. Dostupné z: http://www.cross-ni.com

[4] KROENKE, David. *Databáze*. 1. vyd. Brno: Computer Press, 2015, 496 s. ISBN 978-80-251-4352-0.

[5] Dohledový systém Zabbix. KOLÍSEK, Antonín. *Linuxsoft* [online]. 2013, 11.2.2013 [cit. 2015-04-19]. Dostupné z: http://www.linuxsoft.cz

[6] HOGAN, Brian P. *HTML5 a CSS3: výukový kurz webového vývojáře*. Vyd. 1. Brno: Computer Press, 2011, 272 s. ISBN 978-80-251-3576-1.

[7] STANÍČEK, Petr. *Kompletní průvodce CSS: kaskádové styly*. Vyd. 1. Brno: Computer Press, 2003, 178 s. ISBN 80-722-6872-4.

[8] RYBIČKA, Jiří. *Latex pro začátečníky*. 3. vyd. Brno: Konvoj, 2003, 238 s. ISBN 80-730-2049-1.

[9] STONES, Richard a Neil MATTHEW. *Linux: začínáme programovat*. Vyd. 1. Překlad Jan Škvařil. Praha: Computer Press, 2000, xxxviii, 897 s. Programování. ISBN 80-722-6307-2.

[10] NÁBĚLEK, Zdeněk. *Monitorování aktivních síťových prvků pomocí webové služby*. Ostrava, 2012. Bakalářská práce. VŠB – Technická univerzita Ostrava. Vedoucí práce Ing. Libor Michalek, Ph.D.

[11] KEOGH, James Edward a Mario GIANNINI. *OOP bez předchozích znalostí: průvodce pro samouky*. Vyd. 1. Brno: Computer Press, 2006, 222 s. ISBN 80-251-0973-9.

[12] LACKO, Luboslav. *ORACLE*. Vyd. 1. Brno: Computer Press, 2003, 464 s. ISBN 80-722-6699-3.

[13] LECKY-THOMPSON, Ed a Steven D NOWICKI. *PHP 6: programujeme profesionálně*. Vyd. 1. Překlad Ondřej Gibl. Brno: Computer Press, 2010, 718 s. Programujeme profesionálně. ISBN 978-80-251-3127-5.

[14] SHINDER, Debra Littlejohn. *Počítačové sítě: nepostradatelná příručka k pochopení síťové teorie, implementace a vnitřních funkcí*. Praha: SoftPress, 2003, 752 s. ISBN 80-864-9755-0.

[15] BARRETT, J. *SSH: kompletní průvodce*. Vyd. 1. Brno: SoftPress, 2003, 556 s. ISBN 80-722-6852-X.

A Přílohy

Jelikož je program celkem objemný, není žádoucí zde vypsat veškeré vytvořené třídy, funkce a kódy. Pro základní představu byly na ukázku vybrány následující čtyři soubory.

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>2.0</version>
    <date>2015-03-20T11:00:00Z</date>
    <hosts>
        <host>
            <host>ag-p123456</host>
            <name>ag-p123456</name>
            <proxy/>
            <status>0</status>
            <ipmi_authtype>-1</ipmi_authtype>
            <ipmi_privilege>2</ipmi_privilege>
            <ipmi_username/>
            <ipmi_password/>
            <templates>
                <template>
                    <name>TemplateOffice</name>
                </template>
            </templates>
            <groups>
                <group>
                    <name>POSTOFFICE</name>
                </group>
                <group>
                    <name>R_VC</name>
                </group>
                <group>
                    <name>T_NT</name>
                </group>
            </groups>
            <interfaces>
                <interface>
                    <default>1</default>
                    <type>1</type>
                    <useip>1</useip>
                    <ip>10.10.10.10</ip>
                    <dns/>
                    <port>10050</port>
                    <interface_ref>if1</interface_ref>
                </interface>
                <interface>
                    <default>1</default>
                    <type>2</type>
```

```
            <useip>1</useip>
            <ip>10.10.10.10</ip>
            <dns/>
            <port>161</port>
            <interface_ref>if2</interface_ref>
        </interface>
    </interfaces>
    <applications/>
    <items/>
    <discovery_rules/>
    <macros>
        <macro>
            <macro>{$IF_IN_MAX}</macro>
            <value>4194304</value>
        </macro>
        <macro>
            <macro>{$IF_OUT_MAX}</macro>
            <value>4194304</value>
        </macro>
        <macro>
            <macro>{$OID_IF_IN}</macro>
        <value>.1.3.6.1.2.1.2.2.1.10.40</value>
        </macro>
        <macro>
            <macro>{$OID_IF_OUT}</macro>
            <value>.1.3.6.1.2.1.2.2.1.16.40</value>
        </macro>
    </macros>
    <inventory>
        <inventory_mode>1</inventory_mode>
        <type/>
        <type_full/>
        <name/>
        <alias/>
        <os/>
        <os_full/>
        <os_short/>
        <serialno_a/>
        <serialno_b/>
        <tag/>
        <asset_tag/>
        <macaddress_a/>
        <macaddress_b/>
        <hardware/>
        <hardware_full/>
        <software/>
        <software_full/>
        <software_app_a/>
        <software_app_b/>
```

<software_app_c/>
<software_app_d/>
<software_app_e/>
<contact/>
<location/>
<location_lat/>
<location_lon/>
<notes/>
<chassis/>
<model/>
<hw_arch/>
<vendor/>
<contract_number/>
<installer_name/>
<deployment_status/>
<url_a/>
<url_b/>
<url_c/>
<host_networks/>
<host_netmask/>
<host_router/>
<oob_ip/>
<oob_netmask/>
<oob_router/>
<date_hw_purchase/>
<date_hw_install/>
<date_hw_expiry/>
<date_hw_decomm/>
<site_address_a/>
<site_address_b/>
<site_address_c/>
<site_city/>
<site_state/>
<site_country/>
<site_zip/>
<site_rack/>
<site_notes/>
<poc_1_name/>
<poc_1_email/>
<poc_1_phone_a/>
<poc_1_phone_b/>
<poc_1_cell/>
<poc_1_screen/>
<poc_1_notes/>
<poc_2_name/>
<poc_2_email/>
<poc_2_phone_a/>
<poc_2_phone_b/>
<poc_2_cell/>

```
                <poc_2_screen/>
                <poc_2_notes/>
            </inventory>
        </host>
    </hosts>
</zabbix_export>
```

Výpis 16: Vygenerovaný XML soubor připravený pro import

```
<?php

include 'Begin.php';
require 'Class/CheckUser.php';
$checkUser = new CheckUser();

// Telefonni cislo ISDN linky
if (IsSet($_GET['isdn']) && IsSet($_GET['loc']) && IsSet($_GET['lanIp']) && IsSet(
    $_GET['dns'])) {

    $isdn = $_GET['isdn'];
    $localita = $_GET['loc'];
    $lanIp = $_GET['lanIp'];
    $dns = $_GET['dns'];
}

ob_start();
echo '<h2>Provádím cinknutí na ISDN číslo: <b></b>' . $isdn . '</h2>
    <blink>Čekejte, dojde k přesměrování zpět</blink><br><br>';

$ip_serveru = "10.10.10.10";
$ssh_klic = "/home/user/dir";
$ssh_uzivatel = "user";
$prikaz = "./cinkni " . $isdn;

exec("ssh -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i " .
    $ssh_klic . " " . $ssh_uzivatel . "@" . $ip_serveru . " '" . $prikaz . "'", $datasrv);

ob_flush();
echo "<meta http-equiv='refresh' content='20;vysledek.php?loc=" . $localita . "&ip=" .
    $lanIp . "&dns=" . $dns . "'>";
ob_end_flush();

include 'End.php';
?>
```

Výpis 17: Skript používání pro vzdálené vyžádání ISDN spojení

```
<?php
```

```
class DatabaseLocal {

    /**
     * Parametry pripojeni do databaze k Localhostu
     */
    private $DB_pocitac_local = 'localhost';
    private $DB_Databaze_local = 'localDb';
    private $DB_Jmeno_local = 'user';
    private $DB_Heslo_local = 'pass';
    private $connect;
    private $query;
    private $resultArray;

    /**
     * Funkce pro pripojeni k databazi Localhost
     * @global type $connect
     */
    public function __construct() {
        $this->connect = mysql_connect($this->DB_pocitac_local, $this->
            DB_Jmeno_local, $this->DB_Heslo_local) or die("<h3>Nepodarilo se
            pripojit k databazi</h3>");
        mysql_query("SET CHARACTER SET utf8");
    }

    /**
     * Funkce pro provedeni queryu v databazi Localhost
     * @param type $query
     */
    public function setQuery($query) {
        $this->setDatabase();
        mysql_query("SET CHARACTER SET utf8");
        $this->query = mysql_query($query, $this->connect) or die("<br><h4>
            Dotaz nebyl proveden. NeoÄŤekÄ›vanÄ› chyba.<br></h4>");
    }

    /**
     * Funkce pro nastaveni databazi Localhost
     * @param type $query
     */
    public function setDatabase() {
        mysql_select_db($this->DB_Databaze_local, $this->connect);
    }

    /**
     * Funkce pro navraceni hodnoty queryu v databazi ITSM
     * @param type $query
     */
    public function getQuery() {
        return $this->query;
```

```
    }

    /**
     * vysledky usporadabne do pole
     * @return array
     */
    public function getResult() {
        $this->resultArray = mysql_fetch_array($this->query);
        return $this->resultArray;
    }

    /**
     * vysledky usporadabne do pole
     * @return array
     */
    public function getResultOutageIs() {
        $pole_sloupcu = array();
        for ($i = 0; $i < mysql_num_fields($this->query); $i++) {
            array_push($pole_sloupcu, mysql_field_name($this->query, $i));
        }
        return $pole_sloupcu;
    }

    /**
     * Funkce pro odpojeni od databaze Localhost
     * @global type $connect
     * @return type
     */
    public function __destruct() {
        if ($this->connect) {
            return;
        } else {
            return MYSQL_Close($this->connect);
        }
    }

}

?>
```

Výpis 18: Ukázka třídy DB a funkce které jsou využívány

```
<?php

class snmpWalk {

    private $oidNamePort = '1.3.6.1.2.1.2.2.1.2.';
    private $oidAllPort = '1.3.6.1.2.1.2.2.1.2';
    private $oidMask = '1.3.6.1.2.1.4.20.1.3';
```

```
private $oidIndex = '.1.3.6.1.2.1.4.20.1.2.';
private $oidPortDesc = '.1.3.6.1.2.1.31.1.1.1.18.';
private $oidNamePortEndnumber;
private $communityString = 'heslo';

public function __construct($ip) {

    $_SESSION['All'] = Array();
    $this->getMask($ip);
    $this->getIndex($ip);
    $this->getName($ip, $this->oidNamePortEndnumber);
    $this->getDesc($ip, $this->oidNamePortEndnumber);
    $this->drawTable();
    $this->getAllPort($ip);
}

public function __destruct() {
    unset($_SESSION['Mask']);
    unset($_SESSION['Ip']);
    unset($_SESSION['Oid']);
    unset($_SESSION['Name']);
    unset($_SESSION['Desc']);
}

private function getMask($ip) {
    $_SESSION['Mask'] = Array();
    $command = 'snmpwalk -c ' . $this->communityString . ' -v 1 ' . $ip . ' ' .
        $this->oidMask;
    $exec = exec($command, $output);
    for ($index = 0; $index < count($output); $index++) {
        $_SESSION['Mask'][$index] = $output[$index];
        $this->parseIpAndMask($output[$index], $index);
    }
}

private function getIndex($ip) {
    $_SESSION['Oid'] = Array();
    $command = 'snmpwalk -c ' . $this->communityString . ' -v 1 ' . $ip . ' ' .
        $this->oidIndex;
    $exec = exec($command, $output);
    for ($index = 0; $index < count($output); $index++) {
        $_SESSION['Oid'][$index] = $output[$index];
        $intOid = explode(' ', $output[$index]);
        $this->oidNamePortEndnumber = $this->oidNamePortEndnumber .
            $intOid[3] . ';';
        $this->parseOid($output[$index], $index);
    }
}
```

```
private function getName($ip, $arrayIndex) {
    $_SESSION['Name'] = Array();
    $array = explode(';', $arrayIndex);
    for ($index = 0; $index < count($array); $index++) {
        $command = 'snmpwalk -c ' . $this->communityString . ' -v 1 ' . $ip . ' ' .
            $this->oidNamePort . $array[$index];
        $exec = exec($command, $output);
        $_SESSION['Name'][$index] = $exec;
        $this->parseName($exec, $index, $array[$index]);
    }
}

private function getDesc($ip, $arrayIndex) {
    $_SESSION['Desc'] = Array();
    $array = explode(';', $arrayIndex);
    for ($index = 0; $index < count($array); $index++) {
        $command = 'snmpwalk -c ' . $this->communityString . ' -v 1 ' . $ip . ' ' .
            $this->oidPortDesc . $array[$index];
        $exec = exec($command, $output);
        $_SESSION['Desc'][$index] = $exec;
        $this->parseDesc($exec, $index, $array[$index]);
    }
}

private function getAllPort($ip) {
    echo '<h2>Výpis všech nalezených interface :<br></h2>';
    echo '<center><table width="20%" border="1"><tr><th>Název IF</th></tr>'
        ;
    $_SESSION['Oid'] = Array();
    $command = 'snmpwalk -c ' . $this->communityString . ' -v 1 ' . $ip . ' ' .
        $this->oidAllPort;
    $exec = exec($command, $output);
    for ($index = 0; $index < count($output); $index++) {
        $_SESSION['Oid'][$index] = $output[$index];
        $interface = explode(' ', $output[$index]);
        echo '<tr><td>' . $interface[3] . '</td></tr>';
    }
    echo "</table></center>";
}

private function parseIpAndMask($snmpStringGetMask, $index) {
    $string1 = explode(' ', $snmpStringGetMask);
    $string2 = explode('.', $string1[0]);
    $ip = $string2[1] . '.' . $string2[2] . '.' . $string2[3] . '.' . $string2[4];
    $mask = $string1[3];
    $_SESSION['All'][$index]['Ip'] = $ip;
    $_SESSION['All'][$index]['Mask'] = $mask;
}
```

```
    private function parseOid($snmpStringGetMask, $index) {
        $string1 = explode(' ', $snmpStringGetMask);
        $oid = $string1 [3];
        $_SESSION['All'][$index]['Oid'] = $oid;
    }

    private function parseName($snmpStringGetMask, $index, $oid) {
        $string1 = explode(' ', $snmpStringGetMask);
        if (isset($string1 [3]) ) {
            $name = $string1[3];
            $_SESSION['All'][$index]['Name'] = $name;
        } else {
            $_SESSION['All'][$index]['Name'] = 'Nenalezeno';
        }
    }

    private function parseDesc($snmpStringGetMask, $index, $oid) {
        $string1 = explode(' ', $snmpStringGetMask);
        if (isset($string1 [3]) ) {
            $desc = $string1 [3];
            $_SESSION['All'][$index]['Desc'] = $desc;
        } else {
            $_SESSION['All'][$index]['Desc'] = 'Nenalezeno';
        }
    }

    private function drawTable() {
        echo '<center><table width="60%" border="1"><tr><th>Název IF</th><th>
            IP</th><th>Maska</th><th>OID</th><th>Popis</th><th>Info</th></
            tr>';
        for ($index = 0; $index < count($_SESSION['All']); $index++) {
            if ($_SESSION['All'][$index]['Name'] == 'Nenalezeno') {
                ;
            } else {
                echo "<tr><td>" . $_SESSION['All'][$index]['Name'] . "</td><td>" .
                    $_SESSION['All'][$index]['Ip'] . "</td><td>" . $_SESSION['All'][
                    $index]['Mask'] . "</td><td>" . $_SESSION['All'][$index]['Oid'] . "
                    </td><td>" . $_SESSION['All'][$index]['Desc'] . "</td><td></td
                    ></tr>";
            }
        }
        echo "</table></center>";
    }

}

?>
```

Výpis 19: Třída využívaná pro získávání informací z routeru